Spring 5-10-2014

Activity-Aware Sensor Networks for Smart Environments

Debraj De

TITLE: ACTIVITY-AWARE SENSOR NETWORKS FOR SMART ENVIRONMENTS

by

DEBRAJ DE

Under the Direction of Prof. WenZhan Song

ABSTRACT

The efficient designs of Wireless Sensor Network protocols and intelligent Machine Learning algorithms, together have led to the advancements of various systems and applications for *Smart Environments*. By definition, *Smart Environments* are the typical physical worlds used in human daily life, those are seamlessly embedded with smart tiny devices equipped with sensors, actuators and computational elements. Since human user is a key component in Smart Environments, human motion activity patterns have key importance in building sensor network systems and applications for Smart Environments. Motivated by this, in this thesis my work is focused on *human motion*

activity-aware sensor networks for Smart Environments. The main contributions of this thesis are in two important aspects: *(i)* Designing event activity context-aware sensor networks for efficient performance optimization as well as resource usage; and *(ii)* Using binary motion sensing sensor networks' collective data for device-free real-time tracking of multiple users.

Firstly, I describe the design of our proposed event activity context-aware sensor network protocols and system design for Smart Environments. The main motivation behind this work is as follows. A sensor network, unlike a traditional communication network, provides high degree of visibility into the environmental physical processes. Therefore its operation is driven by the activities in the environment. In long-term operations, these activities usually show certain patterns which can be learned and effectively utilized to optimize network design. In this thesis I have designed several novel protocols: *(i) ActSee* [1] for activity-aware radio duty-cycling, *(ii) EAR* [2] for activity-aware and energy balanced routing, and *(iii) ActiSen* [3] complete working system with protocol suites for activity-aware sensing/ duty-cycling/ routing.

Secondly, I have proposed and designed *FindingHuMo* [4] (Finding Human Motion), a Machine Learning based real-time user tracking algorithm for Smart Environments using Sensor Networks. This work has been motivated by increasing adoption of sensor network enabled Ubiquitous Computing in key Smart Environment applications, like Smart Healthcare. Our proposed *FindingHuMo* protocol and system can perform device-free tracking of multiple (unknown and variable number of) users in the hallway environments, just from non-invasive and anonymous binary motion sensor data.

INDEX WORDS: Smart environments, Wireless sensor networks, Routing, Radio duty-cycling, User tracking, Machine learning.

TITLE: ACTIVITY-AWARE SENSOR NETWORKS FOR SMART ENVIRONMENTS

by

DEBRAJ DE

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

in the College of Arts and Sciences

Georgia State University

2014

TITLE: ACTIVITY-AWARE SENSOR NETWORKS FOR SMART ENVIRONMENTS

by

DEBRAJ DE

Committee Chair: WenZhan Song

Committee: Sushil Prasad

Xiaolin Hu

Mark Keil

Electronic Version Approved:

Office of Graduate Studies

College of Arts and Sciences

Georgia State University

May 2014

DEDICATION

This dissertation is dedicated to Georgia State University. This dissertation is also dedicated to my parents Alok Ranjan De and Atreyi De for their endless support, sacrifice, hard work, love and passion for academics.

ACKNOWLEDGEMENTS

This dissertation work would not have been possible without the support of many people. I want to express my gratitude to my advisor Professor WenZhan Song for his continuous guidance, patience and support. It is not only his invaluable academic knowledge and methodologies, but also his passionate attitude and discipline to succeed in my future career development. He gave me the invaluable passion and effort for quality research.

I'd also like to thank my fellow lab-mates and co-workers for helping me through valuable knowledge sharing and contributions. I made many great friends and co-workers at Georgia State University, Washington State University, Ohio State University during my PhD and MS endeavors. I'll especially thank Mingsen Xu and Lei Shi for their wonderful and memorable companionship with research works, productive discussions, support, and real great friendships.

Finally, I wish to thank all of my family members and all my friends for their unconditional support, love, patience and understanding.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

- GSU - Georgia State University
- CS - Computer Science
- WSN - Wireless Sensor Network
- ATPG - Activity Transition Probability Graph
- HMM - Hidden Markov Model
- RFID - Radio-frequency identification
- MAC - Medium Access Control
- CMDP - Constraint Markov Decision Process
- LQI - Link Quality Indicator
- RSSI - Received Signal Strength Indicator
- MCU - Microprogrammed Control Unit
- GPS - Global Positioning System
- PIR - Passive Infrared Sensor
- EATG - Extended Activity Transition Graph

PART 1

INTRODUCTION

In this chapter we first introduce the main background of the thesis: *human motion activity-aware Wireless Sensor Networks for Smart Environments*. Then we present the main research challenges and our solution approach.

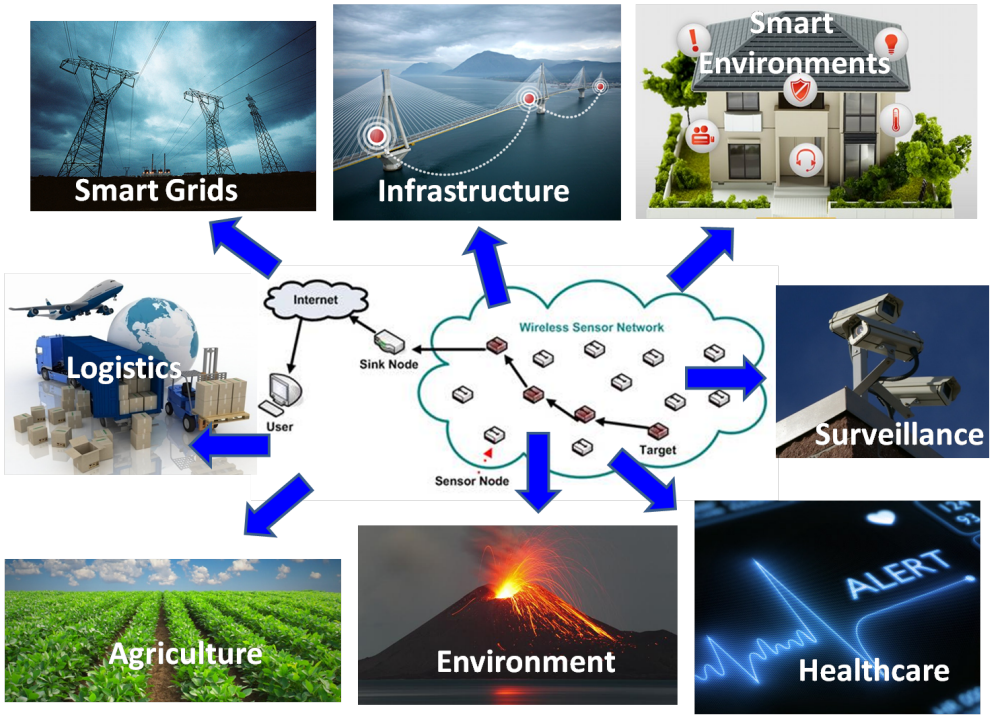


Figure 1.1 Example of application domains of Sensor Networks.

1.1 Introduction

Wireless Sensor Networks consist of spatially distributed devices communicating through wireless radio and cooperatively sensing spatial-temporal physical or environmental conditions. They provide a high degree of visibility into the environmental physical processes. Wireless Sensor Networks (largely termed as *WSN*) have been used in pervasive domains of applications such

as: ubiquitous computing in Smart Environments, healthcare, scientific exploration, infrastructure protection, military and surveillance, social, assisted living, and many more (indicated in Figure 1.1). However, there are significant challenges to the design for sustainability and reliability for real-world applications. Those challenges come in all forms: software, hardware and application specific designs. The important components of Wireless Sensor Networks (WSN) system are as shown in Figure 1.2.

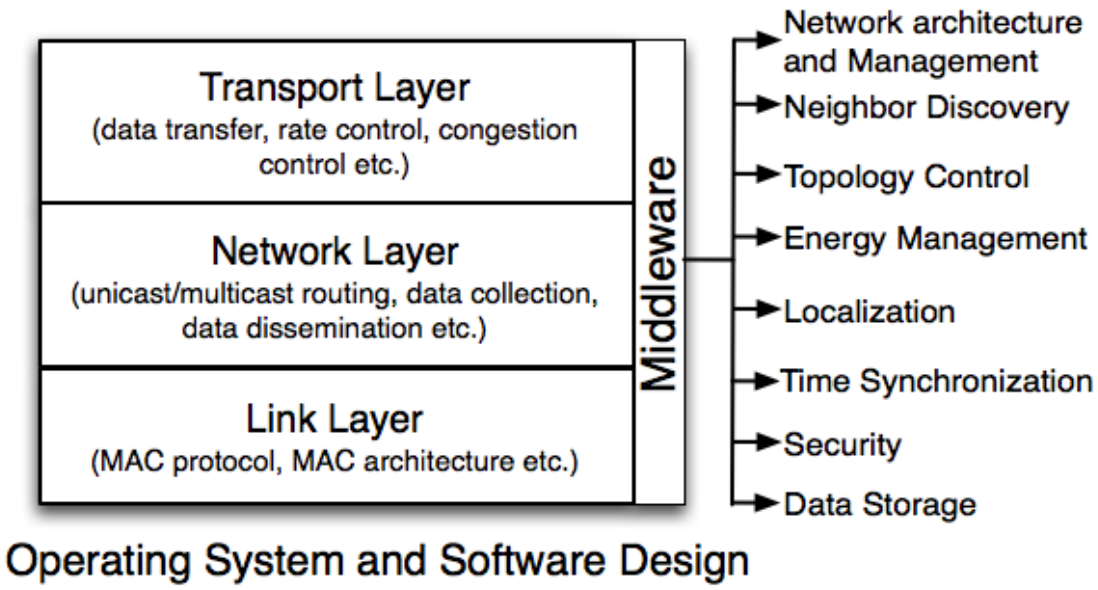


Figure 1.2 Different layers and components of Wireless Sensor Networks.

1.1.1 Smart Environments

One of the most emerging application domain for Sensor Networks is the *Smart Environments*. The formal definition of Smart Environments can be as follows. *Smart Environments* are varied physical worlds typically used in human daily life, those are seamlessly embedded with tiny devices capable of pervasive sensing, actuating and computing. These physically embedded tiny devices are all connected through a continuous network for data collection, in order to enable various pervasive applications and services. The Smart Environments include *Smart Home*, *Smart Building*, *Smart Workplace*, *Smart Farm*, *Smart Clinic*, *Smart Meeting Room* etc. Example scenarios of Smart Environments are show in Figure 1.3 and Figure 1.4. With the help of *Wireless Sensor*

Networks and *Machine Learning* technologies, *Ubiquitous Computing* is gaining momentum in *Smart Environments* applications.

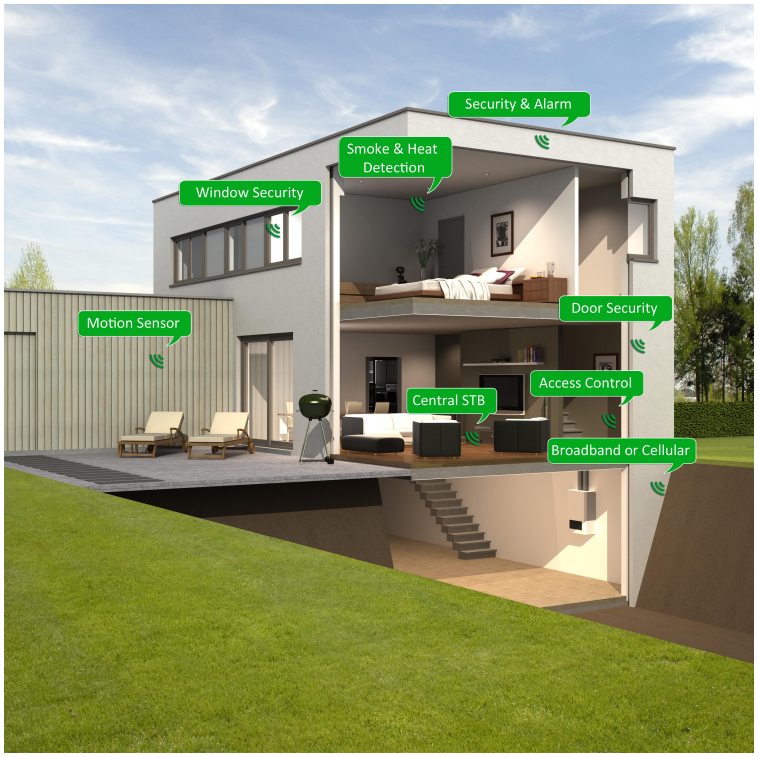


Figure 1.3 Smart Home example scenario.

1.1.2 Motivation and Challenges

The *Smart Environments* in majority are the physical environments used in human daily life. Therefore human motion activity (spatial and temporal) patterns and behaviors are important factors for both: application or service, and sensor network system design. Motivated by this, in this thesis I have explored two perspectives: *(i) activity-aware system design perspective:* how the human motion activity patterns can be intelligently used for sensor network protocol designs, and *(ii) activity-aware application perspective:* how the human motion activities can be non-intrusively identified for motion tracking applications. To note that this tracking application is very much useful for a whole range of future applications of Smart Environments. One of the most important of them is the *Smart Healthcare*. Being able to identify and analyze individual and group human

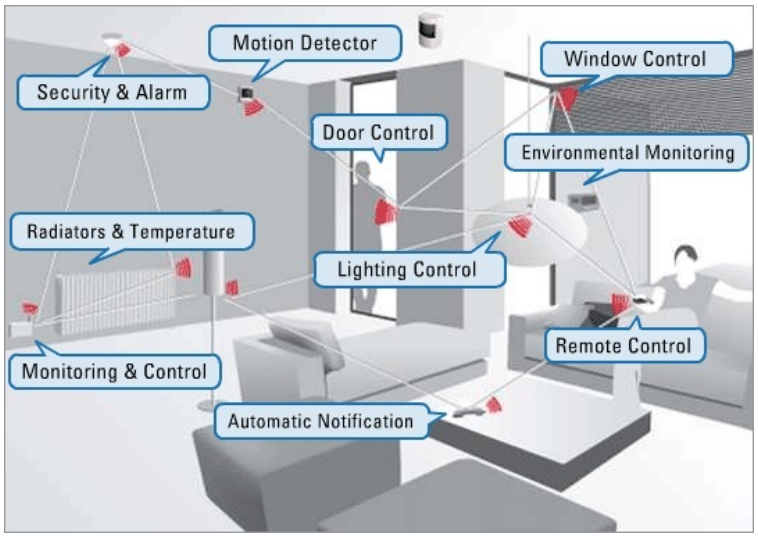


Figure 1.4 Smart Environment example scenario.

motion activity behaviors in regular life in Smart Environments has the great potential in designing the future of remote and proactive *Smart Healthcare* systems.

In this dissertation we I attempt to solve two important research problems: (i) one related to Smart Environments's event activity context-aware sensor network protocol design, and (ii) another related to a specific application of user movement tracking with motion sensor networks in Smart Environments. First I discuss the motivation behind attempting these two research problems. Then I will briefly present my contribution to solve these two problems.

The *first problem* I attempt to solve is designing sensing event activity context-aware networking for sensor networks in Smart Environments. A Smart Environment may contain many highly interactive and embedded devices embedded inside it. These devices may be controlled to meet the demands of the environments and applications. While the Smart Environments offer many societal benefits, they also introduce novel and complex challenges for wireless sensor network protocol design.

For example a future Smart Home may be equipped with dozens of wireless connected sensors that aid in ensuring health and safety of its residents or providing building energy efficiency. If these sensors are continuously operating in full-alert mode, they will consume a great deal of energy and bandwidth. The result is an expensive infrastructure that requires constant maintenance to

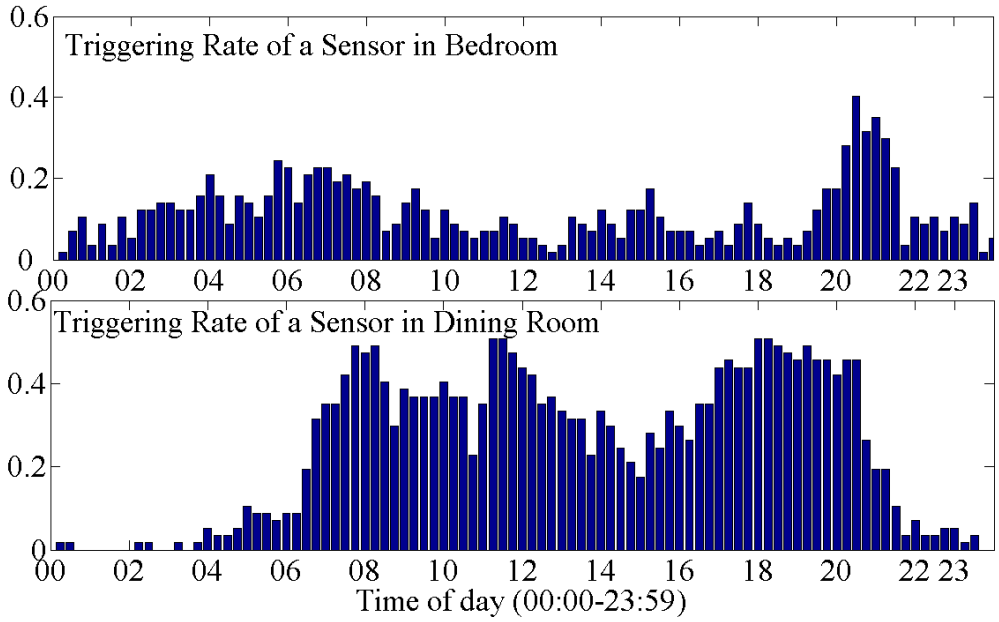


Figure 1.5 Change of activity pattern in 24 hours with time and space

replace batteries in order to meet application performance. Figure 1.5 shows the daylong average triggering rate of two motion sensors (in different location context) in Cairo Smart Home testbed [5]. The triggering rate for two motion sensors, one in bedroom and another in dining room, is calculated using 24 hour data from 57 days. It can be observed that the user movement activity pattern changes in context of time and space. Therefore the sensing and communication resource usage in such sensor network can be adapted using the learned activity pattern. These have motivated us to design an activity context aware sensor network that tries to intelligently adapt the network resources using learned intelligence from activity patterns, without compromising the performance of the application that it serves.

The *second problem* I attempt to solve is designing real-time and scalable multi-user tracking system for any crowded Smart Environments, with the help of simple binary motion detecting sensor networks. Smart Environments are equipped with sensors which keep tracking the movements of users, who can for example be residents in a smart home, or employees in a smart workplace. Modeling the behaviors of users is a key step in developing particular applications in a Smart Environment. Identification and tracking the trajectories of users is the first step towards modeling. In many applications, e.g., in a Smart Workplace, a Smart Clinic, or a Smart Home, users may

not want to reveal their identity all the time. So image or video camera sensors are not applicable in these situations. In addition, the cost of sensors and communication device may drive designers to choose binary sensors that are relatively cost effective and more likely to be accepted by general users. The binary sensors (e.g., a binary proximity based sensor, or a motion detector) only generates binary valued times series motion data. This poses a challenge to identify and track user trajectories of multiple users. Tracking mobile users in Smart Environments has utilization in many effective applications or services such as: data delivery to mobile users indoor ([6]); mobile and social localization ([7]); smart wireless healthcare ([8]) etc. In addition user tracking has application to study the working culture of a workplace [9]. These have motivated us to design a real-time and scalable system for tracking of multiple (unknown and variable number of) targets or human users in crowded Smart Environments.

1.2 Our Approach

Now we briefly describe our contributions to solve the research problems just described above. The details of our research solutions are provided in the chapters to follow.

1.2.1 Activity-Aware Protocols for Sensor Networks

In the *first part* of this work, we present the design of three proposed sensor network protocols: *ActSee*, *EAR* and *Actisen*. *ActSee* [1] is an activity-aware radio duty cycling protocol, given the sensor network can use any routing protocol of it's choice. Then *EAR* [2] is an activity-aware and energy-balanced routing protocol, given the sensor network can use any radio duty cycling protocol. Finally the complete *ActiSen* [3] system is a complete sensor networking solution with activity-awareness integrated in all of: sensing, radio duty cycling and routing. The goal is to imbue wireless sensor networks with cognitive capabilities and activity context awareness, in order to make them act in a more intelligent manner and prolong their lifetime. The proposed and designed protocols use behavioral pattern information from an available probabilistic *activity transition graph* (inferred from activity patterns in the Smart Environment it is deployed in). This knowledge is used to efficiently optimize two seemingly conflicting performance goals (applica-

tion performance and constrained resource usage performance) of the wireless sensor network through: activity-aware sensing, radio duty-cycling and routing.

The operation of the activity-aware sensor network is conceptually different from typical sensor networks in a way that it proactively and adaptively optimizes the network operations using learned activity patterns. The proposed activity-aware protocols and systems are implemented in TinyOS-2.x (one of the most popular operating system for sensor networks). The experimental results from simulation and real testbed experiments with large scale sensor networks indicate that the activity-aware designs of proposed protocols outperform existing designs in: resource optimization performance (energy efficiency, network lifetime etc.) and application performance (event detection, data delivery latency, data delivery throughput etc.) for Smart Environments sensor networks.

1.2.2 Multi-User Tracking with Binary Motion Sensor Network

In the *second part* of this work, we present *FindingHuMo* [4] (*Finding Hu*man *Mo*tion), a real-time user tracking system for Smart Environments. *FindingHuMo* can perform device-free tracking of multiple (unknown and variable number of) users in the Hallway Environments, just from non-invasive and anonymous (not user specific) binary motion sensor data stream. The significance of our designed system are as follows: (a) Fast tracking of individual targets from binary motion datastream from a static wireless sensor network in the infrastructure. This needs to resolve unreliable node sequences, system noise and path ambiguity; (b) Scaling for multi-user tracking where user motion trajectories may crossover with each other in all possible ways. This needs to resolve path ambiguity to isolate overlapping trajectories; *FindingHumo* applies the following techniques on the collected motion datastream: (i) a proposed motion data driven adaptive order Hidden Markov Model with Viterbi decoding (called *Adaptive-HMM*), and then (ii) an innovative path disambiguation algorithm (called $CPDA$). Using this methodology the system accurately detects and isolates motion trajectories of individual users. The system performance is illustrated with results from real-time system deployment experience in a Smart Workplace Environment.

1.2.3 Thesis Organization

The rest of thesis is organized as follows. In chapter 2 we present the related works in the literature. In chapters 3 and 4, we present the proposed *ActSee* and *EAR* protocols respectively. Then in chapter 5 we present the complete activity-aware sensor network system, named *ActiSen*. Next in chapter 6 we present the details of our proposed user tracking algorithm and system called *FindingHuMo*. Finally in chapter 7 we conclude this thesis.

PART 2

RELATED WORKS

In this chapter we discuss the related works in the literature. First we cover the relevant works in sensor networks system deployments in Smart Environments. Then we present related works in key related protocols for sensor networks: radio duty-cycling and routing. Next we discuss the works related to user tracking in Smart Environments.

2.1 Sensor Network Systems for Smart Environments

There is considerable amount of work done on development and deployment of sensor network systems for Smart Environments. BScope [10] presents a sensor network architecture design for activity recognition and analysis. Intelligent and networked sensors enabled in-house monitoring of elders is very much in demand due to considerable increase in aging population. Such service has the great potential of increasing autonomy and independence for the elderly people, while minimizing their risks of living alone. Driven by this need, BScope project designs a scalable framework for detailed behavior interpretation of elders. The system has three main design components: sensors, middleware and behavior interpretation mechanisms. The behavior interpretation mechanisms are designed to analyze and interpret the collected sensor data using a sensory grammar.

ALARM-NET [8] presents the implementation of a wireless sensor network for assisted living and residential monitoring in *Smart Home*. The goal is to improve the quality of healthcare and the prospects of *aging in place* using sensor network technology. This project has attempted solving challenging problems in scalability, energy management, data access, security, and privacy. The system is designed with tiered network architecture with upper layer rich in energy and processing capability, and lower layer with deployed environment sensors and body worn medical sensors. It allows a two-way data flow and data analysis between the front-end and back-end, in order to

enable context-aware protocols designed for the residents' individual patterns of living. This work also discusses the querying system and security issues for smart home sensor network.

The work in [11] uses data from a deployed system to show the vulnerability of daily in-home activity information from a wireless snooping attack, called FATS attack. This work has demonstrated and evaluated the FATS attack on eight different homes containing wireless sensors. Based on the analysis it has proposed and evaluated a set of privacy preserving design guidelines for sensor network systems in *Smart Homes*.

Besides typical sensor nodes, RFID based sensor network [12] is also being used for daily activity recognition. This work uses Intel Wisp RFID sensors emplaced on objects of daily use, for capturing daily activity patterns. Gator Tech Smart House [13] deploys Atlas sensor-actuator platform for behavioral monitoring and alteration for diabetic and obese Individuals. This also presents middleware design in general for smart spaces. The Atlas platform consists a modular architecture for scalable use of sensors and actuators. The work in CASAS [5] and MavHome [14] uses motion and other kinds of sensors for tracking and monitoring the Activities of Daily Living (ADL) for assisted living. Camera sensor network is used in [15] for research on vision-based reasoning for smart environments and ambient intelligence. Other related ongoing works are Smart Medical Home [16], Spinner [17] etc.

But in all smart home sensor networks, activity-awareness is not utilized for optimizing network operation and resources. Context awareness in sensor network is also studied in several works. The work in [18] presents a proactive communication algorithm for context aware sensor network. But activity context awareness for dynamic adaptation of sensor network is an under-utilized research direction. In this thesis we have designed several sensing event activity-aware sensor network protocols (for sensing, radio duty-cycling and routing) and then a complete system, where the sensor network dynamically and proactively adapts the operations to the event activity patterns.

2.2 Duty-Cycling MAC protocols in Sensor Networks

The link layer in sensor network deals with the data transfer among neighboring nodes sharing same wireless link. Due to the lossy wireless communication medium in sensor networks, reliable and fast data exchange necessitates Medium Access Control (MAC). The MAC protocol design in sensor network is required to satisfy some key properties: energy efficiency, scalability to node density, communication synchronization, bandwidth utilization etc. The wireless communication states of sensor nodes, especially the wireless radio idle state is the most energy-consuming operation. This makes efficient design of radio MAC protocol crucially important. There is significant amount of research works done on MAC protocol design for wireless sensor networks. Existing MAC protocols can be categorized into two types: synchronous and asynchronous approaches.

Synchronous MAC protocols specify the period of wake-up and sleep for communication to reduce the unnecessary time and energy wasted in idle listening. Nodes periodically exchange SYNC packets for synchronization and data transfer in the common active schedule. S-MAC [19], T-MAC [20] etc. are examples of synchronous MAC schemes. The other type, the asynchronous MAC protocols have no control overhead for synchronization unlike synchronous schemes, in order to improve the energy efficiency compared. Examples of asynchronous schemes are B-MAC [21], X-MAC [22] etc. They rely on low power listening (LPL), also called channel sampling, to let a sender communicate to a receiver which is duty cycling.

B-MAC utilizes a long preamble to achieve low power communication. In X-MAC, short preambles with target address information are used to reduce the excessive preamble, instead of a long preamble. When a receiver wakes up and detects a short preamble, it looks at the target address that is included in the preamble. If the node is the intended recipient, it keeps awake for the incoming data, otherwise it goes to sleep immediately. Most of these MAC protocols are just based on data traffic generated in the network, and they don't generally learn from event activity patterns and exploit them. So activity context aware radio duty cycling in sensor networks is relatively unexplored and under-utilized.

2.3 Routing Protocols in Sensor Networks

The protocols required in sensor network layer include unicast/multicast routing, data collection and data dissemination. Routing protocols for sensor networks are responsible for maintaining the routes in the network and have to ensure reliable multi-hop communication. Routing in sensor networks is very challenging due to the inherent characteristics (such as many-to-one and one-to-many routing requirements than just one-to-one routing) that distinguish sensor networks from other networks. The unique characteristics of sensor networks require effective methods for data forwarding and processing.

There are some previous works in the literature on activity-aware routing. The work in [23] has proposed an adaptive Randomized Re-Routing (RRR) algorithm, designed to react to congestion caused by unusual activity events in sensor networks so as to provide better quality of service to the packets carrying the novel or unusual activity event data. Some relevant works on energy-balanced or lifetime-maximized routing design issues include [24], [25], [26]. The work in [24] has formulated the lifetime maximization problem as a linear program and has solved it using distributed heuristics technique. But this work assumes that the message generation rate at nodes are fixed and known. In [25] the observation has been made that the linear program is equivalent to that of a maximum concurrent flow problem. The algorithms proposed in [24] and [25] are able to determine how the traffic (generated at a constant rate) should be split among the different routes in order to maximize the network lifetime. Since the traffic generation rates are assumed to be constant and known in these works, the network can solve the energy aware routing problem off-line. The work in [27] converts the maximum network lifetime problem into a utility-based nonlinear optimization problem and proposes a distributed routing algorithm to solve the problem. But this work also assumes that the data generation rate at each node is fixed and known in advance. But sensor networks are majorly driven by activities. Therefore the data generation rates at nodes are usually non-uniform and not known exactly in advance. Our proposed protocol *EAR* uses this more realistic view.

[28] proposes an energy efficient algorithm to find and maintain routes in mobile ad hoc

networks. It borrows the notion of learning from cognitive packet networks (CPN) to design a robust routing protocol. Other recent works on energy aware and other different QoS aware routing protocols are [29], [30], [31]. Self-aware networks are also effective in designing energy efficiency and QoS awareness. Self-aware networks have *self-awareness* through online self-monitoring and measurement, coupled with intelligent adaptive behavior in response to observations. Some latest works on self-aware networks are [32], [33].

For many practical applications (for example smart environment activity detection sensor networks) the message generation rate at different nodes are non-uniform and also dynamic. This problem needs to be solved with online routing protocol which does not need to know the message generation rates. An online routing algorithm max-min zP_{min} is proposed in [26] for network lifetime maximization and it provides a competitive analysis. $CMAX$ [34] proposes an algorithm that tries to maximize the network capacity using shortest path computation with routing metric based on node remaining energy. The work in [35] proposes $E - WME$ online routing algorithm for the scenario of energy harvesting sensor nodes. Most of these energy aware online routing algorithms are based on remaining energy of relaying nodes. Unlike our proposed online routing protocol EAR, these works don't try to maintain energy balance in the network as a whole, and don't learn from activity patterns. In this paper we have considered the issue of energy balance across the network. An energy-welfare index (using Atkinson Inequality Index) is utilized in [36] to keep energy balance in network. But the forwarder selection procedure is complex and expensive (computing the index for each forwarder for each packet). Also there is no theoretical analysis of its proposed routing. To note that, for the goal of maximizing network lifetime one possible solution may be to route the messages along the path with the maximal minimal fraction of remaining energy (the max-min routing). The performance of max-min path can degrade in situations (as described with some example in [26]). Another issue with the max-min routing is that following route with max-min energy node may be expensive compared to other possible paths. For large number of data streams there can be significant energy consumption for common nodes on max-min routing paths. This creates bottleneck nodes with high energy consumption and thus degrades the network lifetime quickly.

There are some works on activity-aware or context-aware networking. The work in [18] presents a proactive communication algorithm for context aware sensor network. A framework for integrated unicast and multicast routing in context-aware ordered meshes is presented in [37]. But utilizing activity awareness for energy efficient and resource optimizing networking is not explored in these works, while this thesis attempts these issues in depth.

2.4 User Tracking in Smart Environments

The problem of tracking multiple targets using sensor networks has been explored by prior references [38], [39], [40] etc. RASS [41] provides a system for transceiver-free user tracking with RF based technology. But it is not suitable when multiple user trajectories overlap. This is because it assumes small enough triangular sized node set deployments to separately detect individual users. The work in [42] uses received signal strength (RSS) measurements for target tracking. Using RSS is unreliable in different physical environments, thus limits it's general applicability. Binary sensors (e.g. passive infrared motion sensors) have drawn considerable contribution ([43], [44] etc) for tracking applications, because of the properties like simplicity, non-invasive property and minimal communication requirements. However, most of the related works are based on some geometric models that can have limited applicability in real-time systems and varying environments. The existing works on multi-target tracking either use expensive and invasive sensors ([45], [46]) or depend on specific models like geometry of sensing range and noise model ([44], [40]).

There are existing works on target tracking using movement modeling based filtering and estimation. They mostly use Bayesian networks [47], Particle filters [48] or Kalman filter [49]. But most of those tracking algorithms in sensor networks either use expensive and invasive sensors (e.g. camera system) or depend on assumed movement model, noise model, sensor calibration, war-driving from WiFi/GSM signals (war-driving is the act of locating and possibly exploiting connections to wireless local area networks while in mobility). For the tracking solution in this thesis, we have used a modified version of Hidden Markov Model (HMM), with in-situ motion activity context. Some existing works on using HMM for multi-target tracking are [50], [46], [51]. But these works use expensive and invasive sensors, while our work uses a proposed activity

context aware HMM with simple binary motion data.

PART 3

ACTSEE: ACTIVITY-AWARE RADIO DUTY CYCLING

In this chapter we present the *Actsee* [1] protocol, an activity-aware radio duty cycling protocol for sensor networks in Smart Environments. *ActSee* utilizes the learned event activity pattern information to intelligently and dynamically adjust radio duty cycles in wireless sensor networks. First we describe in details, the main protocol in *ActSee*. Then we explain the system evaluation and performance analyses of *ActSee*.

3.1 Background

Based on the existing works the following radio duty cycling strategies can be adopted for activity detection sensor network systems. (a) The first strategy is *Uniform duty cycling* that lets each node operate at the same radio duty cycle. However this simple and easy to implement strategy is inefficient under activity context aware environment. This is because it fails to leverage the underlying activity transition pattern. For example in Smart Home, during early morning when the Smart Home residents are expected to be waking up from sleep, it is not necessary to keep all the sensors active with higher duty cycle, but only those sensors which are located in the bedroom, and nodes on active route from them. (b) The second strategy is *Reactive duty cycling*. Only when a node successfully detects the present of an activity, it starts to increase the duty cycle of itself and the sensors on the routing path from itself to the sink node (i.e. base station). Otherwise, the nodes operate at a low duty cycle. Obviously, this strategy outperforms the uniform duty cycling in terms of lower data delivery latency and higher energy efficiency. However, the main shortcoming of this "reactive" strategy comes from the delay involved in delivery of data packets indicating detected events ("decision propagation phase"). In particular, it may take longer time to inform all the sensor nodes on the routing path to increase their duty cycle. When the activity transition appears to be frequent, or the network size is large, the decision propagation delay may become

very significant bottleneck to the application performance. This is often unacceptable in critical time sensitive applications such as in real-time tracking or monitoring. (c) The third strategy can be existing synchronous or asynchronous MAC protocols. These classes of MAC protocols reacts, based on the decision making after event activity data is generated in the network. But they don't have intelligence to exploit the typical activity patterns of the environment.

To our knowledge there doesn't exist much work on duty cycling protocol that learns detected activity patterns and then adapts according to it. Motivated by the shortcomings of existing strategies, in this thesis we have proposed efficient duty cycling strategies that tries to achieve two goals, that are apparently hard to satisfy together: low data delivery latency and higher network lifetime. Besides the novelty of activity-awareness, the designed protocols are also unique in achieving both these goals. *ActSee*: (i) maintains high duty cycle for nodes on the routing paths for active and predicted (predicted to be active in next stage or period, according to activity pattern) data sources (for fast and reliable data delivery), (ii) maintains low duty cycle (for energy saving) for potentially idle nodes which are not predicted to be active next, and (iii) has minimal decision propagation delay to the base station. The big challenge for maintaining such non-uniform duty cycling is that, the distribution of duty cycle of the nodes has to dynamically adapt with change of active and predicted data sources.

3.2 Activity-Aware Radio Duty Cycling

3.2.1 System Model

The Activity Transition Probability Graph ($ATPG$) for a smart environment can be constructed based on the observed activity patterns of sensed events and their transitions. In $ATPG$, a node represents a sensor node in the environment and an edge denotes a pair of sensor nodes that can physically be reached directly from each other. These node pairs are connected with a weighted edge in the $ATPG$ graph that denotes transition of activity between them. In this way, we can estimate the probability of transition between two sensor nodes, say x and y, based on the relative frequency of events at x followed by event at y. Figure 3.1 shows the floorplan and layout

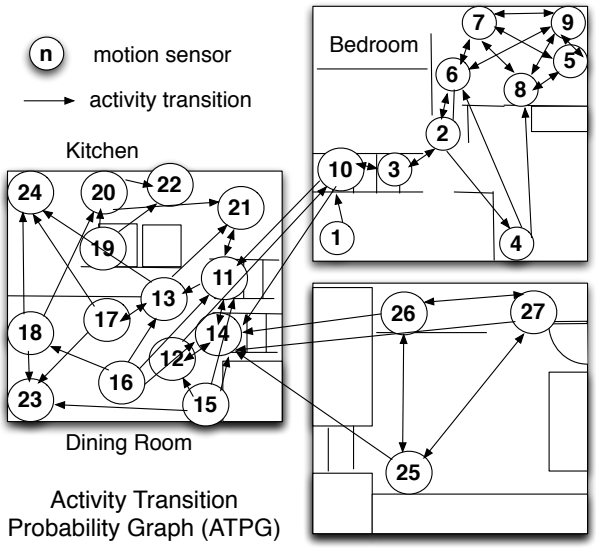


Figure 3.1 Activity Transition Probability Graph ($ATPG$) learnt from the *CASAS* Smart Home testbed [5]. The significant transition probability from node 27 to nodes 14, 25 and 26 are 12%, 45% and 40% respectively.

of sensor node distribution for the $CASAS$ smart home testbed [5]. The weight associated with edge (x, y), denoting the probability of activity transition from x to y, is estimated as:

$$p(x, y) = \frac{number\ of\ events\ for\ x\ followed\ by\ y}{number\ of\ events\ for\ x} \tag{3.1}$$

More specifically, we model $ATPG$ as a discrete time N-state Markov chain, $\mathbf{X} = \{x_1, x_2, \cdots, x_N\}$, with transition probabilities $p(i, j)$ from state x_i to x_j (for $1 \leq i, j \leq N$). Each state x_i is associated with a set of active sensor nodes at that state of activity. For simplicity of presentation, let there be only one active sensor node at each state of activity. In practice, there exists a cluster of active nodes at the same activity state, the cluster can be treated as a single node for analysis. Such activity node clusters can be formed by finding the relatively densely connected component in the activity transition graph. Example of such cluster in a smart home can be the set of nodes in the kitchen, bedroom or bathroom etc. So the nodes closely related to the same activity context usually fall into the same cluster. We denote the sensor node(s) associated with state x_i as v_i. For any state x_i, we define the neighbors of x_i as those states $x_j \in \mathbf{X}$ with positive transition probability $p(i, j) > 0$ from x_i.

The term *state report latency* is defined as the time it takes from the moment an activity enters a new state to the moment the sink node is informed of that activity. In a typical sensor network, the *state report latency* usually includes state detection latency and state delivery latency. The *state detection latency* is defined as the time duration from the moment an event activity (e.g., motion) occurred, to the moment a sensor has detected this event activity. In real-systems the *state detection latency* can be negligible as the new-generation sensor node design achieves wake-on capability [52]. Then *state delivery latency* is defined as the time duration from the moment the activity event has been detected by sensor node to the moment the sink node receives the event data successfully. By assuming the *state detection latency* is zero, in this work the focus is to design an activity-aware radio duty cycling protocol that minimizes the expected *state delivery latency*. Ideally, if the radio works with 100% duty cycle, the activity events can be reported with minimum latency.

Now in the system model, for each node v, we define a finite candidate set for duty cycle assignment $\mathcal{D}(v) = \{d_1, d_2, \cdots, d_n\}$, where $d_i \in \mathcal{D}(v)$. Specifically, $\mathcal{D}(v)$ defines all possible duty cycle assignments for node v. For simplicity of notation, we assume $\mathcal{D}(v) = \mathcal{D}\ \forall\ v$. Typically, $\mathcal{D} = \{2\%, 5\%, 8\%, 10\%, 15\%, 20\%, 25\%\}$. We have formulated the optimal duty cycle assignment problem as a Markov Decision Process, where decisions are made at points of time, referred to as the decision stages or periods. Thus, time is divided into *stages*: $T = \{t | t = 0, 1, 2, \cdots\}$. At the start of each stage, the decision maker observes the system in a state, say $x_i \in \mathbf{X}$, and then chooses action a_j from the set of allowable actions in that state. In this work, an *action* specifies a duty cycle assignment for each sensor node for the next stage.

3.2.2 Validation of Activity State Delivery Latency Model

In this subsection, we validate the estimated formulation for state delivery latency in terms of node hop distance (to sink) and node duty cycle. We have performed experiments in a 100 node sensor network using the *TOSSIM* simulator to compute the state delivery latency (f_L) in terms of hopcount (h) and duty cycle (d). The link-layer model in *TOSSIM* is valid for static and dynamic practical scenarios. From the experimental data as in Figure 3.2 and Figure 3.3, it can be observed that, f_L varies almost linearly with h, and is inversely proportional to d. Therefore, if the routing

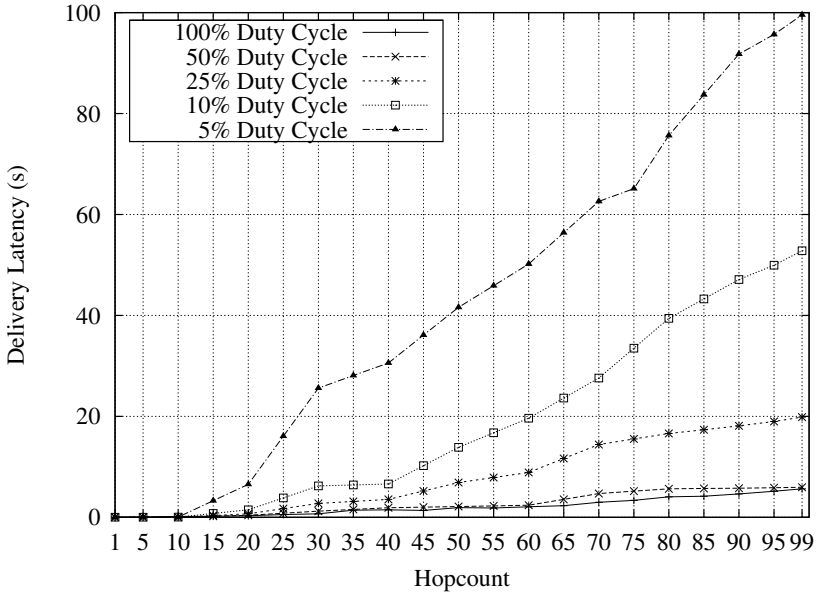


Figure 3.2 State delivery latency (seconds) vs. Hopcount with varying Duty Cycle.

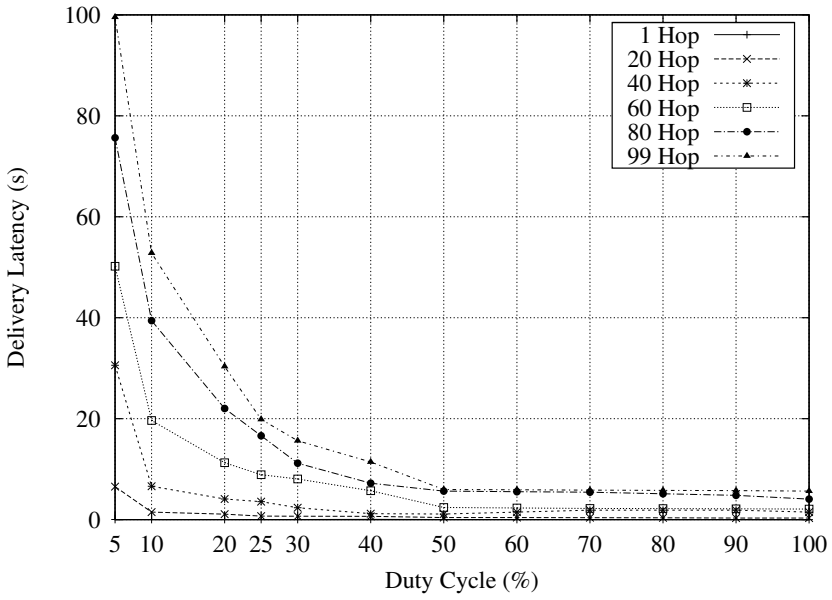


Figure 3.3 State delivery latency (seconds) vs. Duty Cycle (%) with varying hopcount.

path is known and the duty cycle is determined for the nodes on the routing path, the state delivery latency can be estimated as: $f_L \propto \frac{h}{d}$, so $f_L = c \cdot \frac{h}{d}$, where c is a constant factor depending on the system set up factors such as network topology. This estimation of state delivery latency can also be verified theoretically in standard network model. Assuming that all the nodes are synchronized, the nodes on the selected route maintain the same duty cycle d. Then the message can travel $d \cdot T/T_{trans}$ hops during one radio ON interval, where T is the time period of radio duty cycling and T_{trans} is the transmission time for one packet over a link. After that time, the message has to wait during the radio sleep interval in the intermediate forwarder node. Then it is forwarded again via $d \cdot T/T_{trans}$ hops. Since the nodes are strictly synchronized, their radio duty cycle periods align in time. This forwarding of data through radio ON period is performed $\frac{h}{d \cdot T/T_{trans}}$ times before arriving at the sink, where h is the hopcount of sink from the source node. Therefore the total estimated state delivery latency $f_L \propto \frac{h}{d \cdot T/T_{trans}}$. Since T_{trans} is constant for fixed message size and T is also fixed, the formulation of f_L validates our estimated model for data delivery latency: $f_L = c \cdot \frac{h}{d}$. It is worth noting that besides hop distance and duty cycle, collisions and congestions among links may also affect the delivery latency. But in the considered network scenario, collision and congestion are usually negligible. In the network of concern, data traffic is usually low or moderate, e.g., 48 byte packet generation in every 5 seconds during activity detection. At any moment there is only a limited number of active nodes generating data at low rate. Also the wireless radio communication frequency can be chosen so that it does not overlap with other sources of radio signals (e.g., Wi-Fi, microwave) in a smart environment.

3.2.3 Proposed *ActSee* Protocol

In *ActSee*, we define the term E as the *energy budget* which is the maximum expected average duty cycle allowed. We study the following constrained optimization problem: *considering a long state evolving process, given a device energy consumption budget E and an activity state transition matrix P, what is the optimal radio duty cycle assignment strategy μ, such that the expected activity state delivery latency (say, $E[L]$) is minimized, and the expected average duty cycle (say, $E[d]$) is maintained under the budget such that $E[d] \leq E$?*

In the *ActSee* protocol each node keeps track of its hopcount to the sink, as well as the next hop on the route to the sink. No matter what routing policy is used by the system, it will not affect the results derived in *ActSee* which requires any change in routing path and hopcount to the sink to be informed to the system. For ease of explanation, during any state x_i, we partition all the sensor nodes into two disjoint sets: inactive set V_I and active set V_A, where V_A contains all the sensor nodes that are associated with active node x_i and its neighbor states in ATPG graph, as well as those nodes appearing on the routing paths from them to the sink. (If x_i has a self loop, x_i is also the neighbor state of itself.) Essentially, the active set V_A contains all possible sensor nodes that may become active or help in relaying in the next stage or time period. The rest of the nodes belong to V_I. The main idea behind *ActSee* is to increase the duty cycle of V_A in the next stage, while keeping V_I in low duty cycle.

Now to eliminate the decision propagation delay existing in reactive strategy, *ActSee* works as follows. Based on the available nodes in the network and the collected information about activity patterns in form of $ATPG$ graph, a back-end system (connected to sink) runs a linear program routine to select the action set (duty cycle assignments for the nodes in V_A for each possible active node). Note that for continuous events like motion, all the neighbor nodes in ATPG are also the neighbors in communication topology (but not necessarily vice-versa). Now for a deployed network, the back-end system calculates only once, the action set of each possible active node. This action set information is disseminated once, stored in the nodes, and updated when necessary. Once a node is active, it reconfigures the duty cycles of its neighbors and also far-off nodes. It is worth noting that *ActSee exploits the existing beacon message* in a routing protocol to piggyback the duty cycle assignment information. Thus it saves energy for distributed duty cycle assignment task. Until some node is dead or some new node is added, the back-end system does not need to recompute or disseminate the action set. Otherwise it recomputes the action set based on periodically collected regular node status information. The activity pattern typically repeats in smart environments after a reasonable learning period. Thus the linear program routine to select the action set is not required to be run often. After the learning phase, *ActSee* conserves energy in a long period, since it is computed based on the knowledge of activity patterns in $ATPG$.

At the beginning of each stage, the currently active sensor nodes v_i use their duty cycle set for assigning duty cycles to nodes in V_A for the *next* stage. The detailed selection of a proper duty cycle assignment is explained in the next subsection based on the strategy μ for duty cycle selection. Then the decision is propagated to V_A immediately during the current stage. By executing the decision propagation phase during current stage, *ActSee* is able to reduce the decision propagation delay. For the remaining sensor nodes which do not receive any updated duty cycle information, they operate at the lowest duty cycle during the next stage to save energy.

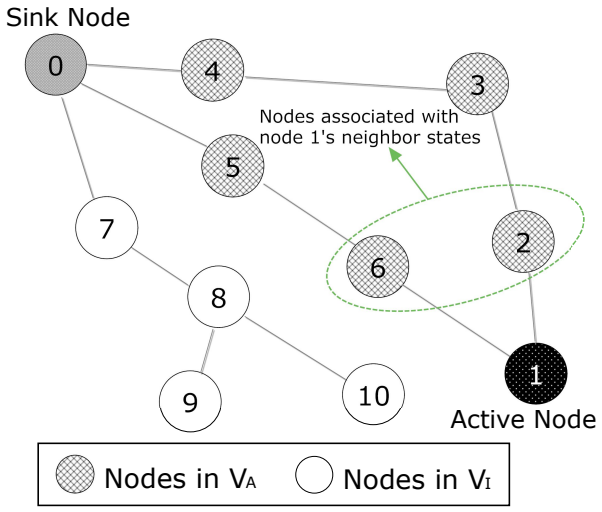


Figure 3.4 Illustration of *ActSee*: node 0 is the sink, node 1 is currently active node, node 2 and node 6 are neighbors of nodes 1. The routing path from node 2 to sink is 2→3→4→0, and that one from node 6 to sink is 6→5→0. In this example, V_A =2, 3, 4, 6, 5, and the set V_I contains the remaining nodes. In *ActSee*, node 1 will pre-select a duty cycle assignment for all nodes in V_A and propagate its decision to them during current stage. All the remaining nodes will work with lowest duty cycle.

Let us illustrate with the example in Figure. 3.4 in which node 1 is the currently active node, while nodes 2 and 6 are neighbors of node 1 in the *ATPG*. More precisely, the states of both 2 and 6 are neighbors of 1's state in the *ATPG*. In this example, the routing path from node 2 is 2→3→4→0, and that from node 6 is 6→5→0. Also $V_A = \{2, 3, 4, 6, 5\}$ and $V_I = \{7, 8, 9, 10\}$. In *ActSee*, node 1 will pre-select a duty cycle assignment for all nodes in V_A and will propagate its decision to them during the current stage, while all the remaining nodes will work at lowest duty cycle. Furthermore, an active sensor node can make a decision immediately based on local

Algorithm 1 Pseudo code for *ActSee* in each node v

Input: Optimal duty cycling strategy μ (computed from Algorithm 2) and current state information
Output: The duty cycle assignment during next stage

1: **if** v does not receive any updated duty cycle information **then**
2: Keep itself in lowest duty cycle in the next state;
3: **if** $v = v_i$ **then**
4: Randomly choose a duty cycle assignment (based on duty cycling strategy μ calculated from Algorithm 2) for all sensor nodes in V_A for the next stage;
5: Propagate the decision to the rest of the nodes in V_A immediately;
6: **if** $v \neq v_i$ and it receives the decision from v_i **then**
7: Adjust duty cycle correspondingly in the next stage;

information, given it knows its action set. Algorithm 1 provides the pseudo code for node. Here, μ specifies the duty cycle assignment of the nodes at each state.

We model the computation of optimal duty cycling strategy problem as a Constraint Markov Decision Process (CMDP). By solving the corresponding Linear Program (LP) in polynomial time, we obtain an optimal strategy μ for each state. Additionally, the selection of different duty cycle for each state is randomized under fixed distribution.

Constraint Markov Decision Process

Markov decision processes (MDP), also known as controlled Markov chains, constitute a basic framework for dynamically controlling systems that evolve in a stochastic way. In a standard MDP, the current action may also affect the transition probability for the next time period, but this is not the case in this paper since the transition graph does not depend on the current duty cycle algorithm. MDP is a generalization of (non-controlled) Markov chains, and many useful properties of Markov chains carry over to controlled Markov chains. The model and problem that we consider in this paper is especially challenging in the sense that more than one objective cost exist, and the controller minimizes one of the objectives subject to constraint on the other.

To apply the above to our problem scenario, we define a tuple $\{O, \mathbf{X}, \mathcal{A}, \mathcal{P}, L, D\}$, where $O = \{t|t = 1, 2, \cdots\}$ denotes the set of decision epochs (note that decisions are made at the beginning

of each stage), and $\mathbf{X} = \{x_1, x_2, \cdots, x_N\}$ is a countable state space. Although we limit our study to discrete activity state transitions, the continuous case can also be handled by dividing it into discrete space. $\mathcal{A}$ is a metric set of actions. We denote $\mathcal{A}(x_i) = \{a_1^i, a_2^i, \cdots\}$ as the action set allowable at state x_i. Each action $a_j^i \in \mathcal{A}(x_i)$ defines a duty cycle assignment for a sensor node in the *next* stage. Let $d(a_j^i, v)$ denote the duty cycle assignment for sensor node v under action a_j^i. Theoretically, each sensor node has $|\mathcal{D}|$ possible duty cycle assignments, thus the action space could be as large as $N^{|\mathcal{D}|}$. In order to reduce the search space, we again leverage the underlying transition graph to facilitate our study. Specifically we restrict the action set $\mathcal{A}(x_i)$ as follows: (i) Only the nodes in V_A, the possible active nodes in the next state, will be considered to increase the duty cycle. The rest of the nodes operate with lowest duty cycle by default. (ii) All relay nodes that appear on the routing path from the same source node, have the same duty cycle as that of their source. (iii) For those nodes appearing on the crossover point of multiple routing paths, *ActSee* sets their duty cycle to the maximum one among all crossing paths.

We use the previous example in Figure 3.4 for illustration. In this example, a possible action a_j^1 selected by node 1 could be $d(a_j^1, 2) = d(a_j^1, 3) = d(a_j^1, 4) = 10\%$ and $d(a_j^1, 6) = d(a_j^1, 5) = 15\%$. Clearly, the size of searching space is $2^{|\mathcal{D}|}$ where 2 is the number of neighbor states of the current state in this example. Now let $\rho(x_i, a_j^i)$ denote the "occupation measure" of state x_i and action a_j^i. It denotes the probability that such state-action pair ever exists in the decision process. $E_v(d)$ denotes the expected average duty cycle of sensor node v, which is expressed as in Eq. (3.2). Notice that the "occupation measure" $\rho()$ is decided by corresponding duty cycling strategy. The term $\mathcal{P}$ are the transition probabilities. We define $\mathcal{P}_{iaj}$ as the probability of moving from system state i to j, when action a is taken. Since different duty cycling strategies will not affect the actual transition process of the event activities, given the activity state transition probability matrix P, it is easy to conclude that $\mathcal{P}_{iaj} = P_{ij}$. Let L be the immediate cost. In this paper, we define $L(x_i, a_j^i)$ as the expected average delivery latency during the next stage by taking action a_j^i, where $L(x_i, a_j^i)$ is expressed as in Eq. (3.3). Recall that in terms of hop count and duty cycle, $f_L(v_k \rightsquigarrow v_0, a_j^i)$ denotes the average delivery latency through a fixed routing path $v_k \rightsquigarrow v_0$ under action a_j^i, and $\mathcal{N}(x_i)$ denotes the neighbor set of state x_i. Then the expected average delivery latency $E[L]$ can be computed as

in Eq. (3.4). E is the maximum allowed expected average duty cycle budget. Therefore for each node v, we have $E_v[d] \leq E$.

$$E_v[d] = \sum_{x_i \in X} \sum_{a_j^i \in A(x_i)} \rho(x_i, a_j^i) \cdot d(a_j^i, v) \tag{3.2}$$

$$L(x_i, a_j^i) = \sum_{x_k \in \mathcal{N}(x_i)} P_{ik} \cdot f_L(v_k \rightsquigarrow v_0, a_j^i) \tag{3.3}$$

$$E[L] = \sum_{x_i \in X} \sum_{a_j^i \in A(x_i)} \rho(x_i, a_j^i) \cdot L(x_i, a_j^i) \tag{3.4}$$

Optimal Duty Cycling Policy μ

In order to compute the optimal strategy of the CMDP with expected average cost criteria, we formulate it as a linear programming (LP) problem. After solving the corresponding linear program, we obtain the optimal strategy through normalization. The following presents how to formulate the duty cycling optimization problem as a linear program.

Problem: *LP-Minimizing Expected Delivery Latency*

Objective: *Minimize* $E[L]$

subject to:

$$\begin{cases} (1)\ \ \rho(x_i, a_j^i) \geq 0, \quad \forall x_i,\ \forall a_j^i \\ (2)\ \ E_v[d] \leq E, \quad \forall v \\ (3)\ \ \sum_{x_i \in X} \sum_{a_j^i \in A(x_i)} \rho(x_i, a_j^i) = 1 \\ (4)\ \ \forall x_j \in X \\ \quad \sum_{x_i \in X} \sum_{a_j^i \in A(x_i)} \rho(x_i, a_j^i)(\delta_{x_j}(x_i) - P_{ij}) = 0 \end{cases}$$

The constraints (1) and (3) ensure that $\rho(x_i, a_j^i)$ is a feasible probability measure. The energy budget can be restricted under the constraint (2) by setting the expected average duty cycle less than E. In inequality (4), $\delta_{x_j}(x_i)$ is the delta function of x_i concentrated on the state x_j.

$$\delta_{x_j}(x_i) = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{otherwise} \end{cases}$$

The constraint (4) describes that the outgoing rate and incoming rate for a state must be the same. At the same time, it emphasizes the property for ergodic processes. After solving the linear program, we get an optimal occupation measure $\rho()$ in terms of delivery latency minimization for each state/action pair. However, since $\sum_{a_j^i \in A(x_i)} \rho(x_i, a_j^i) \leq 1$, we can not directly use $\rho(x_i, a_j^i)$ as the probability of taking action a_j^i at state x_i. Instead, the stationary optimal duty cycling strategy μ can be determined from $\rho(x_i, a_j^i)$ as follows:

$$\mu(a_j^i | x_i) = \frac{\rho(x_i, a_j^i)}{\sum_{a_j^i \in A(x_i)} \rho(x_i, a_j^i)}$$

Here $\mu(a_j^i | x_i)$ describes the probability of taking action a_j^i at state x_i. It is easy to verify that $\sum_{a_j^i \in A(x_i)} \mu(a_j^i | x_i) = 1$. For any number of input states, Algorithm 2 can return an optimal strategy μ in polynomial time. As the input to Algorithm 1, $\mu(a_j^i | x_i)$ for all j will be propagated to each corresponding sensor node v_i.

Algorithm 2 Computation of Optimal Duty Cycling Strategy μ

Input: Energy budget E, transition matrix P, underlying wireless sensor network topology G
Output: Optimal duty cycling strategy μ.

1: Solve corresponding CMDP linear programming to get the occupation measure $\rho(x_i, a_j^i)$, $\forall x_i \in X, \forall a_j^i \in A(x_i)$;
2: Calculate optimal duty cycling strategy μ from $\rho(x_i, a_j^i)$ as:

$$\mu(a_j^i | x_i) = \frac{\rho(x_i, a_j^i)}{\sum_{a_j^i \in A(x_i)} \rho(x_i, a_j^i)}$$

3.3 Experimental Study and Performance Evaluation

The performance of the proposed *ActSee* protocol is evaluated both in (a) a real sensor network testbed, and (b) a sensor network simulator TOSSIM [53]. The experiments with probabilistic event activity transitions are performed with networkwide data collection at the sink node. In the experiments, the active source nodes send activity detection data to the sink node through forwarder nodes on a multi-hop path. In addition to the activity data, each node periodically reports the energy level and other status information to the sink. As the base routing protocol we use the shortest path routing, although *ActSee* can operate with any routing protocol. The whole system is implemented in a manner suitable for real-time applications. The network topology information and event activity data are collected at the sink node, and transferred to the back-end system for $ATPG$ graph generation and solving the Linear Program (using standard method). The back-end system computes the optimal duty cycling strategy, as described in Algorithm 2. Then the resulting action set (duty cycle assignments) from Linear Programming solver, is disseminated back into the network once. The nodes perform the routine described in Algorithm 1. It is important to note that after forming $ATPG$, the duty cycle assignments are disseminated into the network only once. Afterwards, if a node is active, it sets the duty cycle of neighbors and far-off nodes through communication of local beacon message sharing. So until a node dies, or a new node is added, or the activity pattern changes (that can happen only in long time period, typically at least several days, in smart environments), the duty cycle assignment strategy stored in the nodes is not changed. Therefore, the networkwide dissemination of the duty cycle set is performed very rarely. Thus there is minimum communication overhead for setting of duty cycles. The software system is implemented in TinyOS-2.x.

Figure 3.5 shows the sensor node software architecture for $ActSee$. The activities detected by the sensing layer is processed in the application layer. The middleware stores the action set for duty cycle assignment. Now with the current action set, the membership of node (being in the active set V_A or inactive set V_I) and the neighbor set information (from network layer) are transferred to the link layer. Then the node reconfigures its own duty cycle and sends beacon message

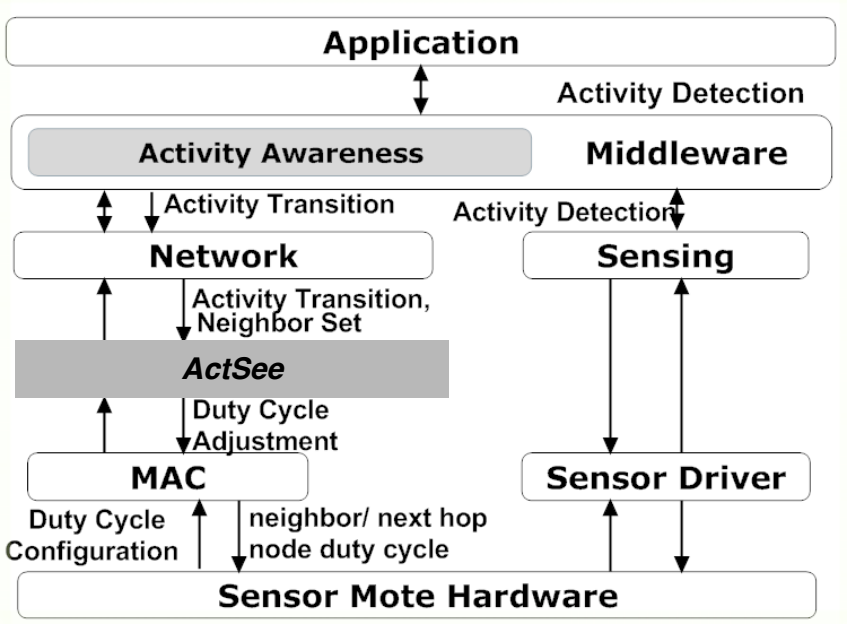


Figure 3.5 TinyOS node software stack with activity-aware design for *ActSee*.

to the neighbors to let them reconfigure their duty cycles accordingly. The energy consumption is calculated using the relevant model of radio transmission, reception and radio idle states. In our experiments, the performance of *ActSee* is compared with *Uniform* duty cycling, *X-MAC* (frequently used in sensor network applications) and *Reactive* strategy. Recall that *X-MAC* represents the class of asynchronous MAC protocols. Moreover, the performance of the Synchronous TDMA MAC protocol performance will be equivalent to *Uniform* where the individual node duty cycles are fixed.

3.3.1 Evaluation on Real Sensor Network Testbed

A network of 16 TelosB sensor motes is deployed on the wall in a smaller area, but according to the physical deployment layout of motion sensor nodes in the kitchen and dining room of *CASAS* smart home [5]. The transmission power of the radio is controlled to generate a multi-hop network in the testbed area. Now light beam is projected in the space, and is programmed to move according to the learned activity transition patterns in $ATPG$. The setup is shown in Figure 3.6 and Figure 3.7. This emulates the detected motion activities of the corresponding smart home

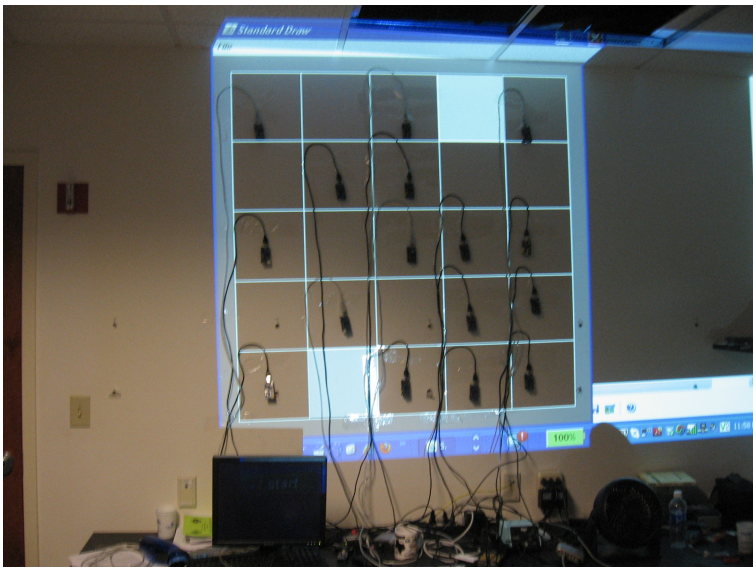


Figure 3.6 Testbed setup.

residents. The standard one-hop broadcast beacon message is utilized in *ActSee* and reactive duty cycling protocols to piggyback extra information to share among the neighbors. A node sensing light intensity above a threshold emulates detected activity. Each experiment with different duty cycling protocols was performed for two hours, in order to generate a variety of probabilistic order of event activities. The experimental set up for comparing different protocols are as follows.

The fixed duty cycle in the *Uniform* strategy is selected at the value to exactly satisfy the previously described *device energy consumption budget* (for achieving at least a minimum network lifetime). In the *Reactive* strategy, the nodes by default maintain minimum duty cycle in the duty cycle assignment set $\mathcal{D}$. But they reactively reconfigure the duty cycle of nodes on an active route to the maximum duty cycle in $\mathcal{D}$. In *X-MAC*, the sleep period is selected in order to maintain a projected average duty cycle satisfying the *device energy consumption budget*. Figure 3.8 shows the distribution of the activity state delivery latency. In *Uniform* and *X-MAC*, around only 50% of the packets are delivered with latency within 9000 ms. In *Reactive*, 70% of the packets are delivered within latency of 9000 ms. Finally, in the *ActSee* protocol, 92% of the packets are delivered within 9000 ms. Therefore *ActSee* provides much reduced data delivery latency.

Figure 3.9 shows instances of delivery latency of 8 consecutive packets delivered at the sink

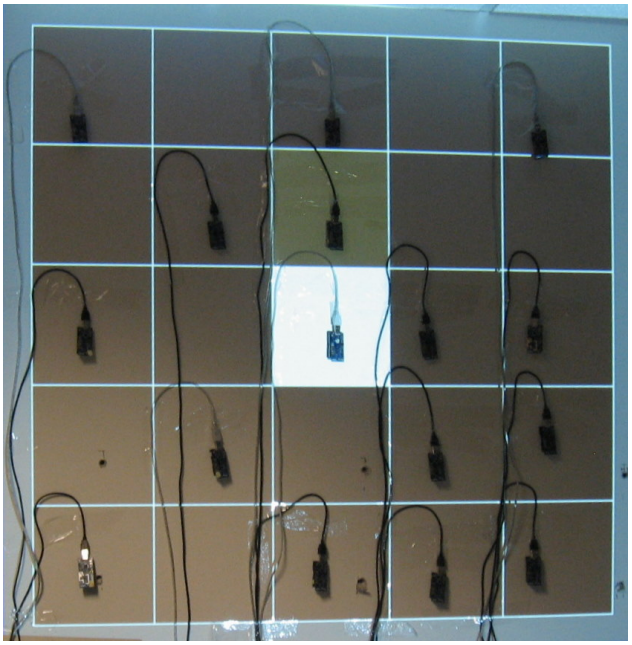


Figure 3.7 Testbed emulating motion activity event with projected light beam.

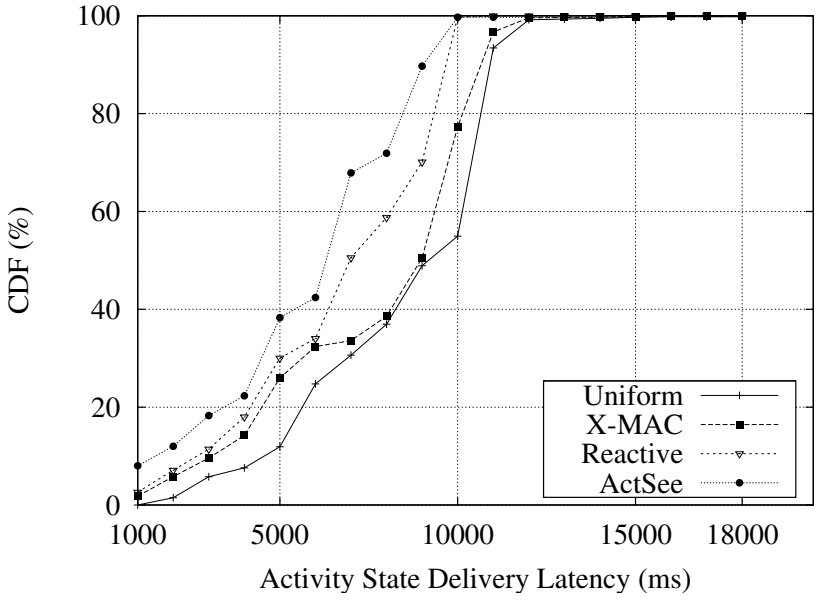


Figure 3.8 Distribution of data delivery latency.

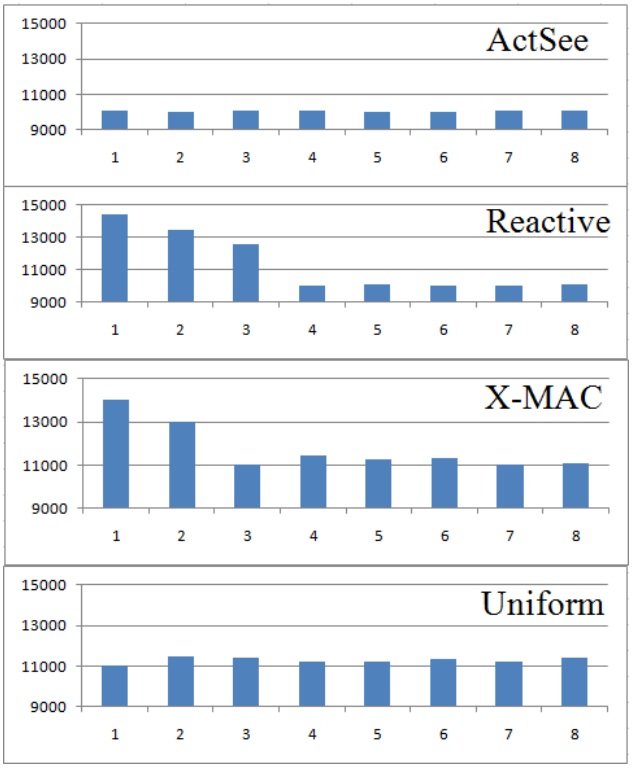


Figure 3.9 Delivery latency of packets after event occurrence for various duty cycling strategies.

after an event occurs. *Uniform* has the same high latency. *X-MAC* has better latency after initial wake-up with preambles, but still suffers from high latency due to insufficient duty cycles of nodes on the active route. *Reactive* performs better from 4th packet, after reactive setting of duty cycles. But *ActSee* maintains low latency for all the packets delivered after the event occurrence. This is very important for time critical applications, where the delivery latency of the very first packet (after event occurrence) is equally or more important than the subsequent ones. *ActSee* achieves this improvement in delivery latency by setting active paths from possible next active nodes to the sink, with high duty cycle. This also leads to better throughput of the collected data at the sink node in *ActSee*, as compared to others. *ActSee* also intelligently saves energy by configuring the idle nodes in the network with low duty cycle. *ActSee* provides the best expected network lifetime (the time between network boot-up and the time when the first node dies).

	Uniform	*X-MAC*	*Reactive*	*ActSee*
Throughput (packets/sec)	7.54	8.52	9.65	12.55
Network Lifetime (days)	91	112	136	221

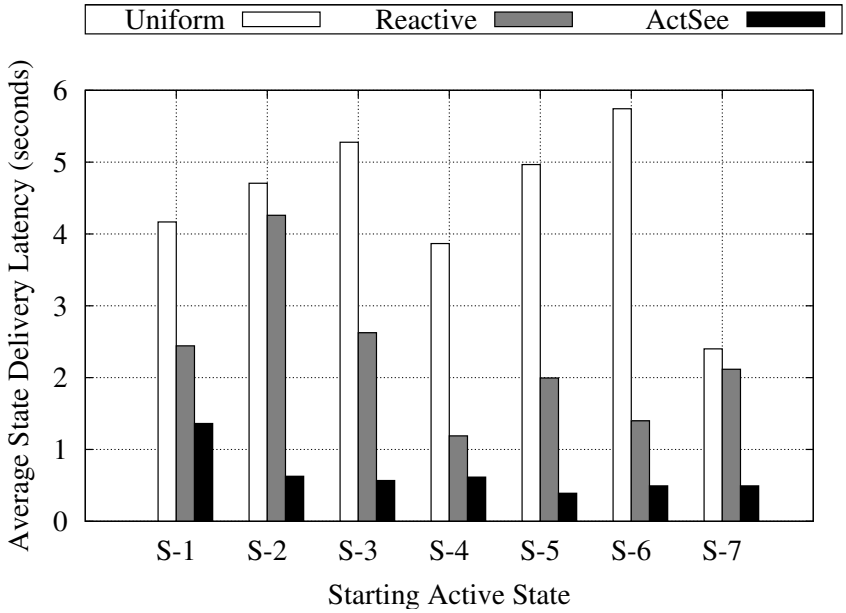


Figure 3.10 State delivery latency (seconds) for different starting active state.

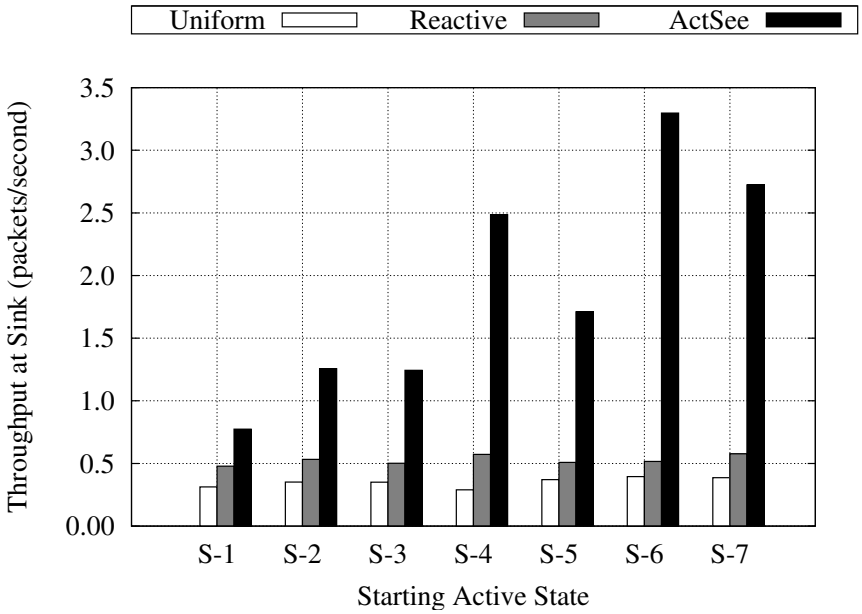


Figure 3.11 Throughput at Sink (packets/second) for different starting active state.

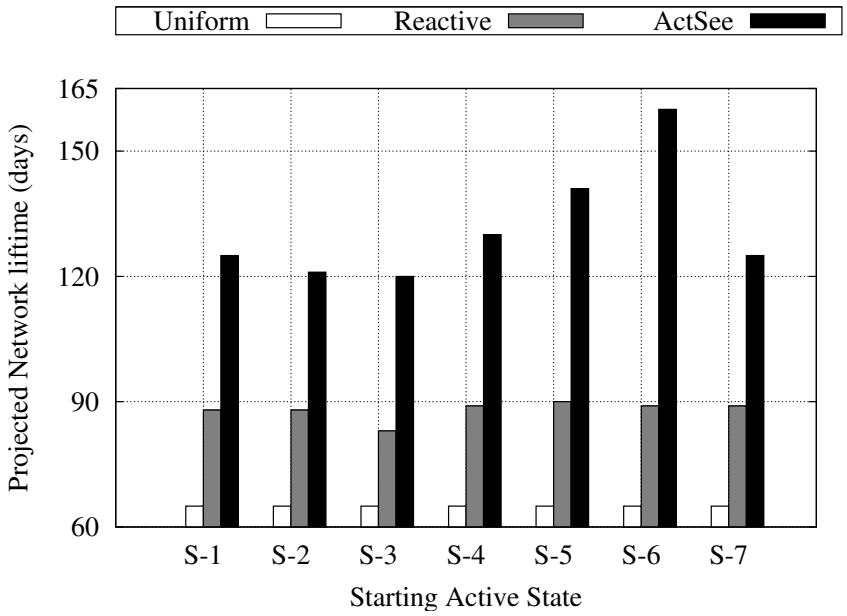


Figure 3.12 Network Lifetime (days) for different starting active state.

3.3.2 Evaluation on Sensor Network Simulator

In the standard sensor network simulator $TOSSIM$ set up, we use a network of 28 nodes in the same layout as the real-time *CASAS* smart home testbed (Figure 3.1). The computed ATPG information is used to generate probabilistic activity transitions among the nodes. Each experiment with different starting states was 60 minutes long, and repeated 20 times to get the average performance and to vary the probabilistic activity transitions.

Figure 3.10, Figure 3.11 and Figure 3.12 show the mean data delivery latency (in seconds), the data throughput at sink (packets/second), and the projected network lifetime (in days) respectively, for each starting state or active node $S-1$, $S-2$, $S-3$, $S-4$, $S-5$, $S-6$ and $S-7$. A state $S-i$ ($1 \leq i \leq 7$) denotes one select node selected from the rooms (kitchen, bedrooms, dining room, etc.) in *CASAS* smart home layout. So if the starting state of activity is different, the order of active states (due to motion of smart home residents) will be different. Our experiments clearly demonstrate that in terms of all the relevant parameters (latency, throughput, lifetime), the proposed protocol *ActSee* significantly outperforms both *Reactive* and *Uniform* duty cycling strategies. For example, the mean state delivery latency in *ActSee* is 67% to 92% better than *Uniform*, and 44% to 85%

better than *Reactive*. The data throughput at the sink in *ActSee* is 2.4 to 8.3 times better than *Uniform*, and 1.6 to 6.4 times better than *Reactive*. Similarly, *ActSee* outperforms others in the network performance metric (e.g., network lifetime). The projected network lifetime in *ActSee* is 1.8 to 2.4 times higher than *Uniform*, and 1.3 to 1.8 times higher than *Reactive*.

3.4 Summary

This chapter presents *ActSee*, a novel activity-aware radio duty cycling protocol for wireless sensor networks. This activity-aware protocol design learns from event activities in smart environments and utilizes knowledge from an activity transition probability graph to dynamically configure the optimal duty cycling strategy in order to provide improved data delivery latency and throughput, while enhancing energy efficiency and hence network lifetime. The experimental results from real sensor network testbed and simulation validate the advantages of the *ActSee* protocol.

PART 4

EAR: ENERGY AND ACTIVITY AWARE ROUTING

In this chapter we present the *EAR* [2] protocol, an energy and activity context aware routing protocol for sensor networks in Smart Environments. *EAR* is an online routing protocol, which chooses the next-hop relay node by utilizing: *activity* pattern information in the *ATPG* graph and a novel index of energy balance in the network. *EAR* extends network lifetime by maintaining an energy balance across the nodes in the network, while meeting application performance with desired throughput and low data delivery latency. First we describe in details, the main protocol in *EAR*. Then we explain the system evaluation and performance analyses of *EAR*.

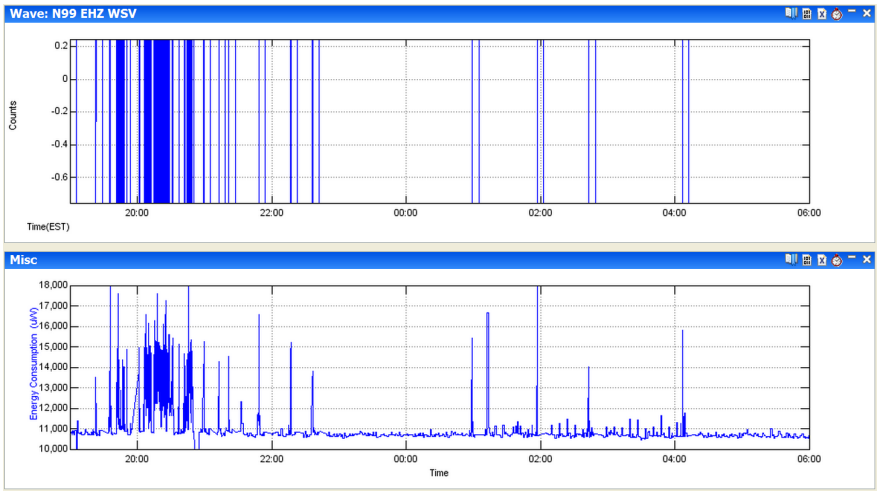


Figure 4.1 Event activity detected and reported by node with motion sensor in smart workplace, and corresponding energy consumption pattern of sensor node.

4.1 Background

Figure 4.1 shows the event activities detected and reported by a node (in a 30 node smart workplace sensor network deployed across a floor) with motion sensor and corresponding node energy consumption. Through observation, it is clear that node energy consumption (thus node

Table 4.1 *List of symbols used*

E_i	Initial energy of node i
$E_i(k)$	residual energy of node i before routing message k
s_k, d_k, l_k	Source, destination and size of message k
c_{ij}	energy required by node i to send unit size data to node j
$p_{tr}(x, y)$	probability of activity transition from node x to y
$t_{tr}(x, y)$	predicted activity transition delay from node x to y
$CL(i)$	activity cluster of node i

operations) is strongly correlated to the event activities. In long-term operations, these activities usually show certain periodic patterns, which can be learned and exploited to optimize network design. However, this has been underexplored in the literature. In this paper we present a novel Energy and Activity aware online Routing (*EAR*) protocol for sensor networks.

4.2 Energy and Activity Aware Routing

We first introduce system models and formal problem definitions, then we describe *EAR* protocol in details.

4.2.1 Preliminaries

The symbols used in *EAR* are listed in Table 4.1.

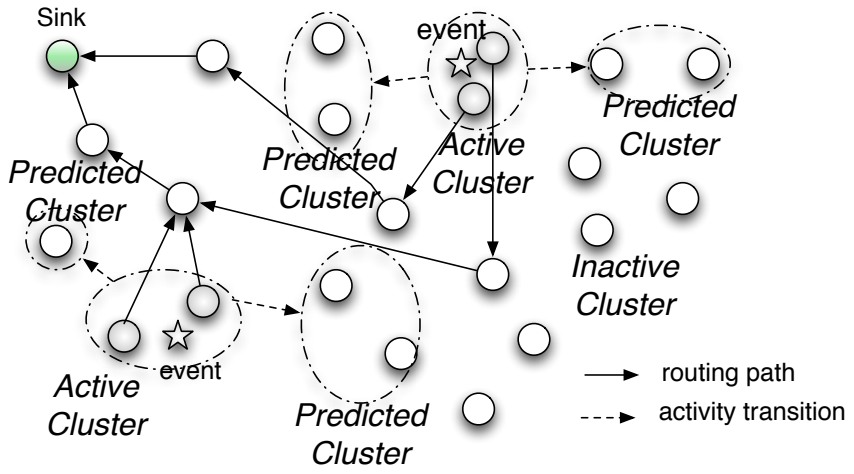


Figure 4.2 Illustration of activity patterns and data generation in network.

Learning Activity Patterns: Figure 4.2 illustrates the event activity patterns and data generation in a sensor network in smart environments. Based on the context of event activities, the nodes in the network at any moment belong to one of the three types of clusters (set of sensor nodes): *Active Cluster* (where the event activities are occurring in current time period), *Predicted Cluster* (predicted to be in active cluster in next time period), *Inactive Cluster* (with no activity in current period and no predicted activity in next time period). The membership of nodes being in clusters changes with time according to an *Activity Transition Probability Graph* (*ATPG*). In such a graph, the edge from node x to node y denotes the transition tuple $< p_{tr}(x,y), t_{tr}(x,y) >$, where $p_{tr}(x,y)$ is the predicted activity transition probability and $t_{tr}(x,y)$ is the predicted activity transition delay, for transition of activity from x-th cluster to y-th cluster. To note that the nodes in *Active Clusters* will be involved in activity detection and computation, followed by sending the data to base station node. Therefore it will be better to avoid involving nodes in Active or Predicted clusters as data forwarder. The nodes with higher probability of being in inactive clusters can be more involved in data forwarding task. This will save some energy (of data forwarding) for active nodes, and will also save some MCU computation resource for the node's own tasks (such as activity detection, processing, communication etc.).

System Model: The energy cost of sleep state is much lower than that of transmitting/receiving state. The energy consumption is considered only for transmitting/receiving state in our system model. The node overhear energy consumption model is used in various earlier works (e.g. in [54]). The sensor network is considered as a graph G=(V, E), where V is the set of nodes and E is the set of edges. Let n=$|V|$ be the number of nodes. Each node i starts with initial energy E_i. The source and destination of message k (of size l_k) are denoted as s_k and d_k respectively. In data collection scenario, d_k is always the base station. Now suppose in multi-hop routing, node i decides to forward message k to node j through link ij. Then node i consumes c_{ij} energy per unit length of data, therefore consuming a total $l_k.c_{ij}$ amount of energy for transmitting message k.

Objective: The design objective of *EAR* is to meet application performance requirement (e.g., throughput and delay) while maximizing network lifetime, by utilizing the activity pattern information in *ATPG* graph.

4.2.2 Algorithm and Protocol Design

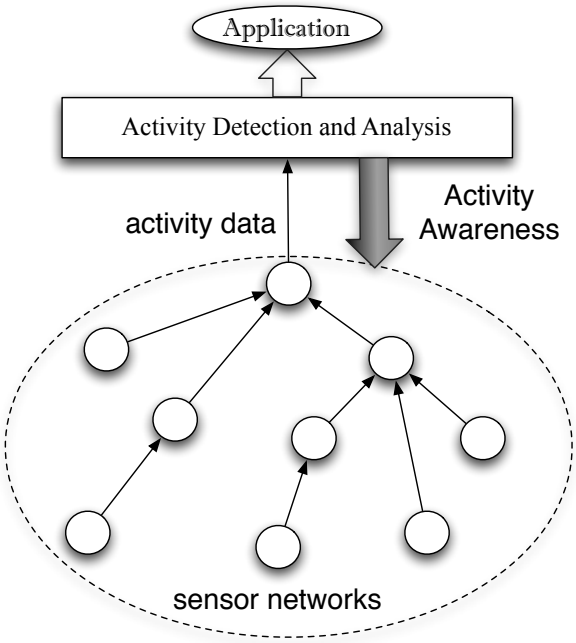


Figure 4.3 Activity-awareness for sensor network.

Distribution of Computation in EAR: The activity-awareness in sensor network is used in *EAR* as shown in Figure 4.3. The event detection data in network is collected at base station node for application purpose, and also used for constructing activity patterns in form of $ATPG$. Then the $ATPG$ information is disseminated back into the network once. Now until a node dies, or a new node is added, or the activity pattern changes (that can happen only in long time period, typically at least several days, in smart environments), the $ATPG$ information stored in the nodes is not changed. Therefore, the networkwide dissemination of $ATPG$ information is performed rarely. Thus there is minimum communication overhead. The activity pattern analysis is only done in base station. All other calculations involved in *EAR* are distributed and localized. So all computations, except activity pattern analysis are distributed and localized in the network.

Building and Maintaining $ATPG$: As a case study, we calculated an $ATPG$ graph based on sensed events in CASAS [5] smart home testbed. The probability of transition between two sensor nodes x and y is based on the relative frequency of events at sensor node x followed by

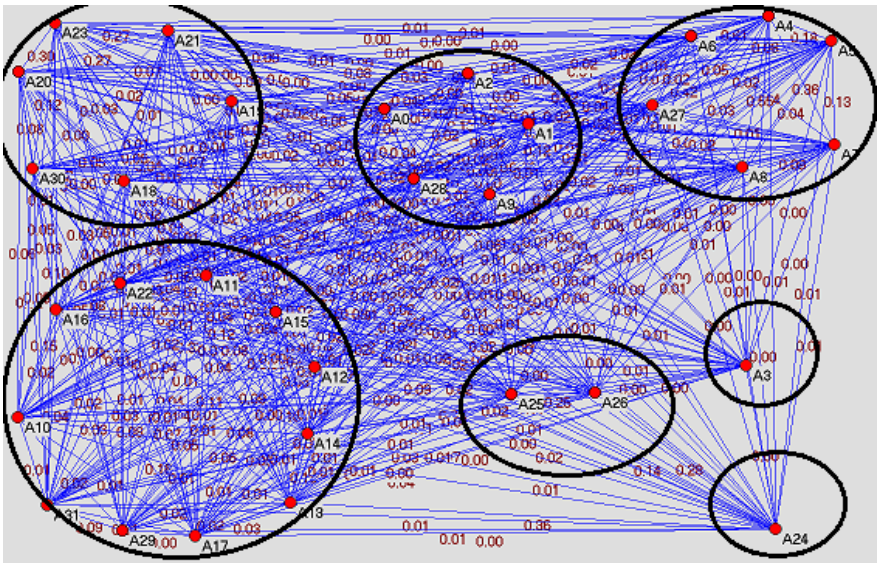


Figure 4.4 Activity Transition Probability Graph (including both significant and negligible transition probabilities) generated from the CASAS Smart Home testbed with layout shown in Figure 4.5.

events at sensor node y. In $ATPG$, a node is generated for each sensor node that exists in the environment. The probability associated with edge x, y is estimated using the formula in equation 4.1. The example $ATPG$ with both significant and negligible transition probabilities is shown in Figure 4.4. The revised $ATPG$ with only significant activity transitions is shown in Figure 4.5.

$$p(x,y) = \frac{|\ events\ for\ sensor\ x\ followed\ by\ sensor\ y\ |}{|\ events\ for\ sensor\ x\ |} \tag{4.1}$$

Activity-Aware Routing Metric: Now we describe the notion of *activity-awareness* in *EAR*. Suppose a node i is trying to forward message k whose source is node s_k, which is in activity cluster $CL(s_k)$. Then node i tries to forward the data to some node j with (a) less computed probability $p(CL(s_k), CL(j))$ of being active (given $CL(s_k)$ is active) and (b) less duration of being active $(t_{active}(j)) \cdot p(CL(s_k), CL(j))$ can be calculated by summing up the computed probability along all the paths from node $CL(s_k)$ to node $CL(j)$ in the $ATPG$. $t_{active}(j)$ can be calculated from weighted (based on transition probability) activity transition delay from $CL(j)$ to the next clusters. Let the period of activity pattern be T_P (which is 24 hours for smart home environments). Then activity-awareness metric for node j when routing of message k is $a(j,k) = p(CL(s_k), CL(j)) \cdot t_{active}(j)$.

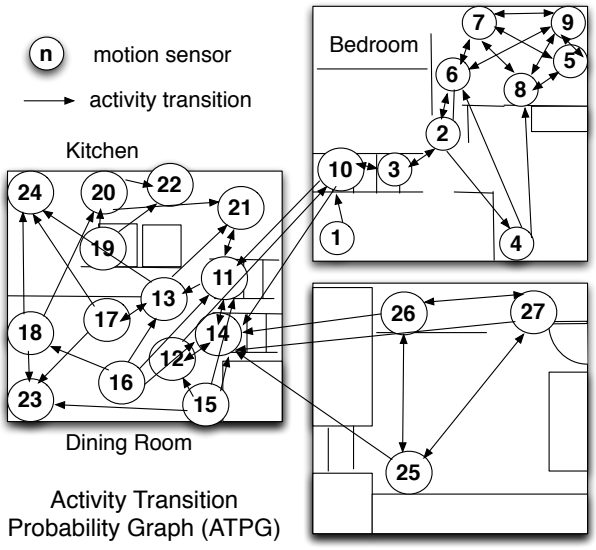


Figure 4.5 Activity Transition Probability Graph (pruned for significant activity transition) learnt from the CASAS Smart Home testbed. The significant transition probability for example from node 27 to nodes 14, 25 and 26 are 12%, 45% and 40% respectively.

$$p(CL(s_k), CL(j)) = \sum_{P \in CL(s_k) \rightsquigarrow CL(j)} \prod_{(xy) \in path\ P} p_{tr}(x, y) \tag{4.2}$$

$$t_{active}(j) = \frac{\sum_{q \in N(CL(j))} p_{tr}(CL(j), q).t_{tr}(CL(j), q)}{T_P} \tag{4.3}$$

Low computation overhead for activity metric: In real application scenario, the nodes don't need to compute the parameters $p(CL(s_k), CL(j))$ and $t_{active}(j)$ each time. The transition graph information (transition probability and duration) can be stored (and updated if necessary in long time duration) in the nodes in an MxM vector, where M is the number of clusters in the network. This will indicate the values of $p(CL(s_k), CL(j))$. Based on that matrix, the nodes can save calculated $t_{active}(j)$ in an 1xM matrix. So the nodes can directly access the routing metric $a(j, k)$. This indicates that Activity Awareness metric doesn't incur much computation overhead.

Energy-Aware Routing Metric: Now we describe the notion of *energy balance* in *EAR*. In order to reach a balance in energy consumption rate across the network we use Atkinson's Inequality Index [55]. It is a measure of economic income inequality in a society. The index can

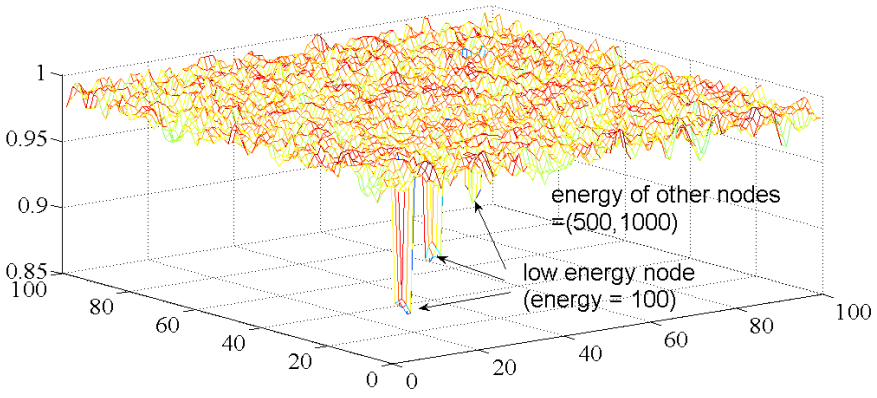


Figure 4.6 Distribution of Energy Balance index (B)

be turned into a normative measure by imposing a coefficient ε to weight incomes. Greater weight can be placed on changes in a given portion of the income distribution by choosing ε, the level of *Inequality Aversion*. The Atkinson index becomes more sensitive to changes at the lower end of the income distribution as ε approaches 1. Conversely, as the level of inequality aversion falls (ε approaching 0) the Atkinson Index becomes more sensitive to changes in the upper end of the income distribution. Atkinson index A is defined as in equation 4.4.

$$A = 1 - \frac{1}{\mu}[\frac{1}{N}\sum_{i=1}^{N} y_i^{(1-\varepsilon)}]^{1/(1-\varepsilon)} \tag{4.4}$$

Where $0 \leq \varepsilon < 1$, y_i is the individual income of i-th entity (i = 1, 2, ..., N) and μ is the mean income of total N entities. We calculate B=$(1 - A)$ as the energy balance index which is computed locally in 1-hop neighborhood. So the index of energy balance $B_i(k)$ computed by each node i (before routing of some message k) is shown in equation 4.5. The term $e_i(k) = E_i(k)/E_i$ denotes the normalized remaining energy before routing message k. $E_i(k)$ is the residual energy of node i before routing message k. The neighbor set of node i is denotes as $N(i)$. Δ (=$\{N(i) \cup i\}$) is the set of 1-hop neighbors of node i and the node itself. So the index B_i is calculated using remaining energy information of the neighbors and the node i itself. In Figure 4.6 the effectiveness of B metric is shown. In a simulated environment in MATLAB with 100x100 sensor network grid, each node has maximum 8 neighbors. Only three nodes in the network have energy value of 100. But

other nodes have energy between 500 to 1000. Then the distribution of locally computed B (with ε=0.8) across the network is shown in Figure 5.10. It can be observed that B is high enough (close to 1) everywhere, except in the neighboring region of the nodes with low energy. Therefore in a distributed network B is a meaningful indicator of region with significant energy imbalance. Lower B indicates higher degree of energy imbalance.

$$B_i(k) = \frac{|\Delta|}{\sum_{j\in\Delta} e_j(k)}[\frac{1}{|\Delta|}\sum_{j\in\Delta} e_j(k)^{(1-\varepsilon)}]^{1/(1-\varepsilon)} \tag{4.5}$$

The nodes in network maintain the hopcount from base station based on the default transmission power level. Now for purpose of routing convergence with delay control, (i) data is forwarded to node with same or less hopcount, (ii) if data is carried by at most H forwarders with same hopcount, it has to be forwarded next to a node with strictly lesser hopcount.

Routing Policy: Let messages are indexed in the order they are generated. Let s_k, d_k and l_k be the source, destination and length of message k. Suppose $E_i(k)$ is the residual energy of node i when the message k is generated but not routed. So $E_i(1)$ is the starting energy E_i for node i. Let the variable $\alpha_i(k)$ = 1-($E_i(k)/E_i$). Variable $B_i(k)$ is the computed energy balance index of node i as described before. $B_i(k)$ denotes the degree of energy balance around the neighborhood of node i before message k is routed. For activity awareness, the previously introduced parameter $a(j,k)$ (= $p(CL(s_k), CL(j)) \cdot t_{active}(j)$) is used where $j \in N(i)$. In the description of the protocol there are two constant parameters λ and σ.

Rationale behind routing policy: Here we explain the rationale behind the routing metric chosen. (a) For the link ij the weight w_{ij} increases with increase in c_{ij} (energy spent by node i for routing message k over link ij). So routing avoids the links with high message transmission cost. (b) w_{ij} increases with increase in both $p(CL(s_k), CL(j))$ (i.e. for nodes with more probability of being active) and $t_{active}(j)$ (i.e. for nodes with more expected active duration). Thus routing tries to select node (as forwarder) with less probability of being active and with less activity duration. This property makes it activity-aware. (c) w_{ij} increases with increase in the energy utilization $\alpha_i(k)$

Algorithm 3 *EAR* algorithm

1. Set the weight or routing metric w_{ij} for the link ij as
w_{ij}=$c_{ij}.a(j,k).(\lambda^{\alpha_i(k)} - 1)/B_i(k)$.
2. Find the best path from s_k to d_k in the graph with the forwarder node selection method described. If node i has data packet to forward, select node j as its forwarder node as follows:

if $hop\_travel < H$ **then**
$j = \arg\min\{w_{iq}, q : q \in N(i)\ AND\ hopcnt(q) \leq hopcnt(i)\ AND\ c_{iq} < E_i(k)/l_k\}$; $if\ hopcnt(j) = hopcnt(i)$ then $hop\_travel+ = 1$;
else
$j = \arg\min\{w_{iq}, q : q \in N(i)\ AND\ hopcnt(q) < hopcnt(i)\ AND\ c_{iq} < E_i(k)/l_k\}$; $hop\_travel = 0$;

3. Let γ_k be the cost of the best path found for message k. Now if $\gamma_k \leq \sigma$, then route the message k along the computed path, otherwise reject it. To note that γ_k=∞ if no such path is found.

of node i. So routing avoids nodes with low normalized residual energy. (d) Then w_{ij} increases with decrease in the energy balance $B_i(k)$ of node i. So routing avoids nodes whose neighborhood is relatively out of balance in residual energy. In addition, to note that in admission control, setting the value of σ to infinity (then the only reasons for rejecting a message is insufficient energy for routing) has shown results with good performance.

Competitive Ratio in Data Delivery: We now describe the calculated competitive bound for *EAR*. Let $c_{max} = max_{ij \in E} c_{ij}$, $c_{min} = min_{ij \in E} c_{ij}$, $a_{min} = min_{ij \in E} a(j,k)$ and ρ=$\frac{c_{max}}{c_{min} a_{min}}$. Let $L(k)$ be the total size of messages that is successfully routed by *EAR* till the arrival of message k. Let $L_{opt}(k)$ be the total size of messages that is successfully routed by optimal offline algorithm till the arrival of message k. Then the obtained competitive ratio result for *EAR* is as shown below. The detailed proof is provided in next section.

Theorem 1 *Suppose* λ=$2(n\rho + 1)$, ρ=$\frac{c_{max}}{c_{min} a_{min}}$, $\sigma = nc_{max}$ *and* Q *is a positive constant. For all message* k, *let*

$$l_k \leq \frac{min_{i \in V} E_i}{c_{max} lg(\lambda)} \tag{4.6}$$

then, $\frac{L(k)}{L_{opt}(k)} \geq \frac{1}{1+Qlg(\lambda)}\ \forall k$

Delivery latency: It can be proved that *EAR* is a H-hop spanner. The H factor assures that the

routed data, carried through less active nodes and energy balanced neighborhood, is converged to the base station. It is important to note that *EAR* actually reduces the data delivery latency to base station, by routing them through less active nodes (nodes less busy with sensing, data processing and forwarding). This is also supported from the experimental results (described later).

Network Lifetime: *EAR* is also proved in following theorem to provide sub-optimal network lifetime.

Theorem 2 *Let T_{ear} and T_{ML} are network lifetime (time till first node dies) for EAR and optimal network lifetime algorithm (algorithm for maximum lifetime) respectively. Then*
$T_{ear} > \frac{T_{ML} \sum_{k=1}^{S} P_{Min}(s_{m_k})}{\sum_{k=1}^{S} P_{atp}(s_{m_k})} + \frac{\delta(\sum_{i=1}^{n} E_i^{ML} - \sum_{i=1}^{n} E_i^{ear})}{\sum_{k=1}^{s} P_{atp}(s_{m_k})}$

S is the number of message generated in the period T_P. $\sum_{k=1}^{S} P_{Min}(s_{m_k})$ is the total energy consumption for routing S message in T_P, when minimum energy path routing scheme is used. $\sum_{k=1}^{S} P_{atp}(s_{m_k})$ is the total energy consumption for routing S message in T_P, when purely activity-aware routing scheme is used. $(\sum_{i=1}^{n} E_i^{ML} - \sum_{i=1}^{n} E_i^{ear})$ denotes the difference between total remaining energy in network after time T_{ML} and T_{ear}. The detailed proof is provided in next subsection.

Reliable Data Delivery: EAR follows routing metric based on energy and activity index. But *EAR* is not affected by link failure rate in lossy wireless medium. It has been observed through a number of experimental works (e.g. in [56]) that for any link, Packet Reception Rate (*PRR*) saturates to sufficiently high (almost 100%) when the link $RSSI$ is at least -90 dBm, or when the Link Quality Indicator *LQI* is 100. In system implementation of *EAR*, a node eliminates its neighbor node from routing table, to whom it's $RSSI$ is < -90dBm or it's *LQI* is < 100. So *EAR* can achieve gains in overall energy and resource usage, while not suffering data delivery guarantee because of failure rate of the links. This makes it practically applicable in any kind of harsh application environment.

Network Energy Balance: Through localized energy balance, *EAR* tries to keep a balance in remaining energy of nodes across network. This is crucial both for networks with uniform and non-uniform (e.g. heterogeneous network) starting energy. This is also useful for energy harvesting sensor networks. Maintenance of energy balance across network inherently increases

lifetime, also gives the opportunity to intelligently utilize dynamically available energy sources. According to [57], Atkinson index measurement of inequality remains unchanged if there is an equi-proportionate change of all levels of income. Now *EAR* ensures messages are not forwarded by nodes with low energy, or not overheard by nodes with very low energy. Thus from the property mentioned, *EAR* tries to thwart the degradation in energy balance in local neighborhood due to routing of messages generating from nodes in non-uniform rate. In this way *EAR* tries to keep better energy balance in the network.

4.3 Theoretical Analysis

We now present the theoretical proof of Theorem 1 and Theorem 2 described in earlier section.

■ **Proof of Theorem 1**: We associate a cost f_i for each node $i \in V$. Now the cost $f_i(k)$ for node i before the arrival of message k is as described in equation 4.7.

$$f_i(k) = E_i(\lambda^{\alpha_i(k)} - 1)/B_i(k) \tag{4.7}$$

Let $S(k)$ be the set of messages those are successfully routed by *EAR* until the arrival of message k. Now to prove the competitive ratio, we first find the lower bound of total message length successfully routed by *EAR*, in terms of node cost f_i.

Lemma 1 $\sum_{i \in V} f_i(k) \leq 2qMP.lg(\lambda).\sigma L(k)$

Proof 1 *Considering any message $k' \in S(k)$, from equation 4.7, for any node $i \in V$:*

$$
\begin{aligned}
& f_i(k'+1) - f_i(k') \\
& \quad \leq \frac{E_i.(\lambda^{\alpha_i(k'+1)} - 1)}{B_i(k'+1)} - \frac{E_i.(\lambda^{\alpha_i(k')} - 1)}{B_i(k')} \\
& \quad = \frac{E_i.\lambda^{\alpha_i(k')}}{B_i(k')}.(\frac{B_i(k')}{B_i(k'+1)}.\lambda^{\alpha_i(k'+1)-\alpha_i(k')} - 1) \\
& \qquad\qquad -\frac{E_i}{B_i(k')}.(\frac{B_i(k')}{B_i(k'+1)} - 1) \\
& \quad \leq \frac{E_i.\lambda^{\alpha_i(k')}}{B_i(k')}.(\frac{B_i(k')}{B_i(k'+1)}.\lambda^{l_{k'}e_{ij}/E_i} - 1) \\
& \qquad\qquad -\frac{E_i}{B_i(k')}.(\frac{B_i(k')}{B_i(k'+1)} - 1)
\end{aligned}
$$

Δ (=$\{N(i) \cup i\}$) is the set of node i and its neighbors. We define two terms $X(k')=\sum_{p\in\Delta} E_p(k')$ and $Y(k')=\sum_{p\in\Delta-i} E_p(k')^{(1-\varepsilon)}$. Then due to cost of routing message k′ for node i and cost of overhearing message k′ by awake neighbors of i:

$$\sum_{p\in\Delta} E_p(k'+1) = X(k'+1) = X(k') - l_{k'}e_{ij} - \beta_1 \tag{4.8}$$

$$\sum_{p\in\Delta} E_p(k')^{1-\varepsilon} = Y(k') + E_i(k')^{1-\varepsilon} \tag{4.9}$$

$$\sum_{p\in\Delta} E_p(k'+1)^{1-\varepsilon} = Y(k') + (E_i(k') - l_{k'}e_{ij})^{1-\varepsilon} - \beta_2 \tag{4.10}$$

β_1 and β_2 are energy cost due to message overhearing, and they vary with every message k′. Now

we compute the expression $\frac{B_i(k')}{B_i(k'+1)}$.

$$\begin{aligned}
&\frac{B_i(k')}{B_i(k'+1)} \\
&\quad = \frac{\frac{|\Delta|}{\sum_{p\in\Delta} E_p(k')}.(\frac{1}{|\Delta|}\sum_{p\in\Delta} E_p(k')^{(1-\varepsilon)})^{1/(1-\varepsilon)}}{\frac{|\Delta|}{\sum_{p\in\Delta} E_p(k'+1)}.(\frac{1}{|\Delta|}\sum_{p\in\Delta} E_p(k'+1)^{(1-\varepsilon)})^{1/(1-\varepsilon)}} \\
&= \frac{(X(k') - l_{k'}e_{ij} - \beta_1).(\frac{1}{|\Delta|}(Y(k') + E_i(k')^{(1-\varepsilon)}))^{1/(1-\varepsilon)}}{X(k').(\frac{1}{|\Delta|}(Y(k') + (E_i(k') - l_{k'}e_{ij})^{(1-\varepsilon)} - \beta_2))^{1/(1-\varepsilon)}} \\
&= \frac{(X(k') - l_{k'}e_{ij} - \beta_1).2^{1/(1-\varepsilon).(lg(Y(k')+E_i(k')^{(1-\varepsilon)}) - lg(|\Delta|))}}{X(k').2^{1/(1-\varepsilon).(lg(Y(k')+(E_i(k')-l_{k'}e_{ij})^{(1-\varepsilon)}-\beta_2) - lg(|\Delta|))}} \\
&= \frac{(X(k') - l_{k'}e_{ij} - \beta_1)}{X(k')}.2^{1/(1-\varepsilon).lg(\frac{Y(k')+E_i(k')^{(1-\varepsilon)}}{Y(k')+(E_i(k')-l_{k'}e_{ij})^{(1-\varepsilon)}-\beta_2})} \\
&= \frac{(X(k') - l_{k'}e_{ij} - \beta_1)}{X(k')} \\
&\qquad .(\frac{Y(k') + E_i(k')^{(1-\varepsilon)}}{Y(k') + (E_i(k') - l_{k'}e_{ij})^{(1-\varepsilon)} - \beta_2})^{1/(1-\varepsilon)}
\end{aligned}$$

Now, the term $T1=\frac{(X(k')-l_{k'}e_{ij}-\beta_1)}{X(k')}$ *is slightly lower than 1, the term* $T2= (\frac{Y(k')+E_i(k')^{(1-\varepsilon)}}{Y(k')+(E_i(k')-l_{k'}e_{ij})^{(1-\varepsilon)}-\beta_2})^{1/(1-\varepsilon)}$ *is slightly higher than 1. This is due to relatively small amount of energy consumption in each routing step (with respect to the remaining energy of nodes). Then it can be proved that* $T1.T2 \leq M$*, where* M *is a relatively high positive constant. Then,* $\frac{B_i(k')}{B_i(k'+1)} \leq M$.

Now from expression of $f_i(k'+1) - f_i(k')$*:* $f_i(k'+1) - f_i(k') \leq 2\frac{E_i.\lambda^{\alpha_i(k')}}{B_i(k')}.(\frac{B_i(k')}{B_i(k'+1)}.\lambda^{l_{k'}e_{ij}/E_i} - 1)$*, and since value of* λ *is high. Therefore:*

$$\begin{aligned}
&f_i(k'+1) - f_i(k') \\
&\quad \leq 2\frac{E_i.\lambda^{\alpha_i(k')}}{B_i(k')}.(\frac{B_i(k')}{B_i(k'+1)}.\lambda^{l_{k'}e_{ij}/E_i} - 1) \\
&\quad \leq 2\frac{E_i.\lambda^{\alpha_i(k')}}{B_i(k')}.(M\lambda^{l_{k'}e_{ij}/E_i} - 1) \\
&\quad = 2\frac{E_i.\lambda^{\alpha_i(k')}}{B_i(k')}.(M2^{l_{k'}e_{ij}lg(\lambda)/E_i} - 1)
\end{aligned}$$

Since $l_k \leq \frac{min_{i \in V} E_i}{c_{max} lg\lambda}$, *therefore* $l_{k'} c_{ij} lg(\lambda)/E_i \leq 1$. *For* $0 \leq x \leq 1$, $2^x \leq (x+1)$. *Therefore:*

$$\begin{aligned} & f_i(k'+1) - f_i(k') \\ & \leq 2\frac{E_i.\lambda^{\alpha_i(k')}}{B_i(k')}.(Ml_{k'}c_{ij}lg(\lambda)/E_i + M - 1) \\ & \leq \frac{2qM.l_{k'}c_{ij}lg(\lambda)\lambda^{\alpha_i(k')}}{B_i(k')} \end{aligned}$$

This is because λ *is very high and q is a relatively large positive constant. Now let* $P(k')$ *be the path over which the message* k' *was successfully routed. Therefore* $\sum_{ij \in P(k')} c_{ij} a(j,k')(\lambda^{\alpha_i(k')} - 1)/B_i(k') \leq \sigma$.

$$\begin{aligned} & \sum_{i \in V} (f_i(k'+1) - f_i(k')) \\ & = \sum_{ij \in P(k')} (f_i(k'+1) - f_i(k')) \\ & \leq \sum_{ij \in P(k')} \frac{2qM.l_{k'}c_{ij}lg(\lambda)\lambda^{\alpha_i(k')}}{B_i(k')} \\ & = 2qM.lg(\lambda)l_{k'} \sum_{ij \in P(k')} \frac{c_{ij}(\lambda^{\alpha_i(k')} - 1)}{B_i(k')} \\ & \quad + 2qM.lg(\lambda)l_{k'} \sum_{ij \in P(k')} \frac{c_{ij}}{B_i(k')} \\ & \leq 4qM.lg(\lambda)l_{k'}\sigma \end{aligned}$$

To note that $|P(k')| < n$. *For* $k' \notin S(k)$, $f_i(k'+1) - f_i(k') = 0$, $f_i(1) = 0 \; \forall i \in V$. *Then:*

$$\begin{aligned} & \sum_{i \in V} f_i(k) \\ & = \sum_{k' \in S(k)} \sum_{i \in V} (f_i(k'+1) - f_i(k')) \\ & \leq \sum_{k' \in S(k)} 4qM.lg(\lambda)l_{k'}\sigma \\ & = 4qM.lg(\lambda)\sigma L(k) \end{aligned}$$

Let $NS(k)$ be the set of messages successfully routed by the optimal off-line algorithm but rejected by EAR, until arrival of message k. Now we show that: $\forall k' \in NS(k)$, $\sum_{ij \in P(k')} c_{ij} a(j,k')(\lambda^{\alpha_i(k')} - 1)/B_i(k') > \sigma$.

A message $k' \in NS(k)$ is rejected by EAR if: (i) there is not sufficient energy on some node to forward the message, or (ii) $\gamma'_k > \sigma$. Now the lemma holds true for situation (ii). We have to prove the lemma for situation (i). Let message k' is rejected due to situation (i) in the protocol. That message k' is successfully routed by optimal offline algorithm through path say $P_{opt}(k')$. But for EAR, there is at least a link $i'j' \in P_{opt}(k')$, for which $E_{i'}(k') < l'_k c_{i'j'}$. Therefore $\alpha_{i'}(k') = 1 - E_{i'}(k')/E_{i'} \geq 1 - (1/lg\lambda)$ (using equation 4.6). Therefore:

$$\begin{aligned}
&\sum_{ij \in P_{opt}(k')} c_{ij} a(j,k')(\lambda^{\alpha_i(k')} - 1)/B_i(k') \\
&\quad \geq c_{i'j'} a(j',k')(\lambda^{\alpha_{i'}(k')} - 1)/B_{i'}(k') \\
&\quad > c_{i'j'} a(j',k')(\lambda^{1-(1/lg\lambda)} - 1)/B_{i'}(k') \\
&\quad = c_{i'j'} a(j',k')(\lambda/2 - 1)/B_{i'}(k') \\
&\quad \geq c_{min} a_{min}(\lambda/2 - 1) = n c_{max} = \sigma
\end{aligned}$$

Finally we show that:

$$n c_{max}(L_{opt}(k) - L(k)) \leq \sum_{i \in V} f_i(k) \tag{4.11}$$

$$
\begin{aligned}
& nc_{max}(L_{opt}(k) - L(k)) \\
& \quad \leq \sum_{k' \in NS(k)} nc_{max} l_{k'} \\
& \quad < \sum_{k' \in NS(k)} \sum_{ij \in P(k')} l_{k'} c_{ij} a(j,k') (\lambda^{\alpha_i(k')} - 1)/B_i(k') \\
& \quad \leq \sum_{k' \in NS(k)} \sum_{ij \in P(k')} l_{k'} c_{ij} f_i(k')/E_i \\
& \quad \leq \sum_{i \in V} f_i(k) \sum_{k' \in NS(k), ij \in P(k')} l_{k'} c_{ij}/E_i \\
& \quad \leq \sum_{i \in V} f_i(k)
\end{aligned}
$$

The second last step uses the fact that the node cost f_i is non-decreasing. Last step uses the fact that the total energy spent for routing the messages at a node cannot exceed its initial energy. Finally from Lemma 1 and equation 4.11, we can prove the following expression, thus proving Theorem 1. ($Q = 4qM$ is a positive value.)

$$\frac{L(k)}{L_{opt}(k)} \geq \frac{1}{1 + Q.lg(\lambda)} \tag{4.12}$$

■ **Proof of Theorem 2**: Competitive ratio analysis implicitly proves the sub-optimality of *EAR* in lifetime w.r.t application point of view. To note that the competitive ratio analysis for *EAR* used no previous knowledge of message arrival. Now for analysis of another definition of network lifetime (time till the first node dies), we have utilized a property that is common to Smart Environment applications. The nodes in sensor networks in such scenario generate same amount of data in each time period, although in each period the data generation sequence may be different. The time period can be short or long. This is actually common to a lots of sensor network applications, for example Smart Home sensor networks, where the daily activity patterns are same, thus message generation is roughly periodic. This is validated through collected motion detection sensor network data in real experiments. So we have assumed here that in each time period $[t, t+\delta)$, the message distributions on the nodes in the network are the same. Then its possible to schedule the message routing with the same policy in each time period of δ.

Now let the network starts at time $t = 0$, network lifetime on optimal routing algorithm (for maximum lifetime) is say T_{ML}, network lifetime on *EAR* routing algorithm is T_{ear}. The initial energy content of each node $i \in V$ is E_i, remaining energy of each node $i \in V$ after time T_{ML} is $E_i(T_{ML})$, remaining energy of each node $i \in V$ after time T_{ear} is $E_i(T_{ear})$. Let the message sequence in any time period is $m_1, m_2, ..., m_{S-1}, m_S$.

$$\sum_{i=1}^{n} E_i = \sum_{i=1}^{n} E_i(T_{ML}) + \sum_{k=1}^{M(T_{ML})} P_{m_k}^{ML} \tag{4.13}$$

$$\sum_{i=1}^{n} E_i = \sum_{i=1}^{n} E_i(T_{ear}) + \sum_{k=1}^{M(T_{ear})} P_{m_k}^{ear} \tag{4.14}$$

Where $M(T_{ML})$ and $M(T_{ear})$ are the number of messages routed from time 0 to T_{ML} and from time 0 to T_{ear} respectively. $P_{m_k}^{ML}$ and $P_{m_k}^{ear}$ are the power consumption of the k-th message m_k by running optimal algorithm for maximum lifetime and *EAR* algorithm respectively. The messages are same in any two periods, without considering the sequence. Therefore it is possible to schedule the messages so that the message rates along the same route are the same in any two periods. Therefore:

$$\sum_{k=1}^{M(T_{ML})} P_{m_k}^{ML} = \frac{M(T_{ML})}{S} \sum_{k=1}^{S} P_{m_k}^{ML} = \frac{T_{ML}}{\delta} \sum_{k=1}^{S} P_{m_k}^{ML} \tag{4.15}$$

$$\sum_{k=1}^{M(T_{ear})} P_{m_k}^{ear} = \frac{T_{ear}}{\delta} \sum_{k=1}^{S} P_{m_k}^{ear} \tag{4.16}$$

$P_{m_k}^{ear}$ is the energy consumption of the message m_k in a period by running algorithm *EAR*. Now *EAR* considers remaining energy, energy balance and activity-awareness. Thus the total energy consumption $\sum_{k=1}^{S} P_{m_k}^{ear}$ will be less than that of ($\sum_{k=1}^{S} P_{m_k}^{act}$) a purely activity-aware routing algorithm (say *act*) that uses routing metric $a(j, m_k)$ for each node j. So, $\sum_{k=1}^{S} P_{m_k}^{ear} < \sum_{k=1}^{S} P_{m_k}^{act'}$. Now, for each message m_k, it is possible to construct the Network Activity Transition Probability graph for the sensor network G. $ATP(s_{m_k})$ is the constructed ATPG graph where the data source

is s_{m_k}, the weight for each node j is $a(j, m_k)$, and $P_{atp}(s_{m_k})$ is the computed energy consumption of greedily selected path from s_{m_k} to base station using the node weight $a(j, m_k)$. So $\sum_{k=1}^{S} P_{atp}(s_{m_k})$ can be computed from G and ATP. Now, $\sum_{k=1}^{S} P_{m_k}^{ear} < \sum_{k=1}^{S} P_{m_k}^{act} = \sum_{k=1}^{S} P_{atp}(s_{m_k})$. On the other hand $\sum_{k=1}^{S} P_{m_k}^{ML} > \sum_{k=1}^{S} P_{Min}(s_{m_k})$, where $P_{Min}(s_{m_k})$ is that of the the minimum energy consumption path in G from s_{m_k} to base station. $P_{Min}(s_{m_k})$ can be computed from G. Therefore:

$$\sum_{i=1}^{n} E_i^{ear} + \frac{T_{ear}}{\delta} \sum_{k=1}^{S} P_{atp}(s_{m_k}) > \sum_{i=1}^{n} E_i^{ML} + \frac{T_{ML}}{\delta} \sum_{k=1}^{S} P_{Min}(s_{m_k}) \tag{4.17}$$

$$T_{ear} > \frac{T_{ML} \sum_{k=1}^{S} P_{Min}(s_{m_k})}{\sum_{k=1}^{S} P_{atp}(s_{m_k})} + \frac{\delta(\sum_{i=1}^{n} E_i^{ML} - \sum_{i=1}^{n} E_i^{ear})}{\sum_{k=1}^{s} P_{atp}(s_{m_k})} \tag{4.18}$$

4.4 Implementation and Performance Evaluation

In this section we have described the implementation, experiments and the analysis of results in detail.

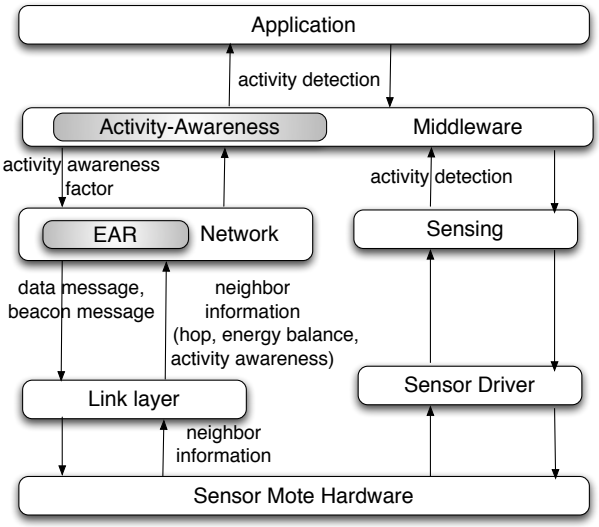


Figure 4.7 TinyOS node software stack included with activity-awareness and energy-balance design for *EAR*.

4.4.1 Implementation of *EAR*

In *EAR* protocol with admission control, the data source node needs to have some knowledge about the network topology and the current energy of nodes. However, in practical network the topology and energy level of the nodes change frequently. It may work for small networks using information dissemination, but will be difficult to maintain for large networks. In this aspect, in our implementation, *EAR* is locally applied to each one hop neighborhood in the network. We have implemented *EAR* and other comparing routing protocols in *TinyOS-2.x*, one of the most popular event based operating system environment for wireless sensor networks. The TinyOS node software stack with activity-awareness and energy-balance design support for *EAR* is shown in Figure 4.7. Regarding activity awareness in the experiments, the set of paths of activity in network is used for probabilistic path selection, and the network nodes are injected with intelligence of corresponding probabilistic transition information (*ATPG* graph).

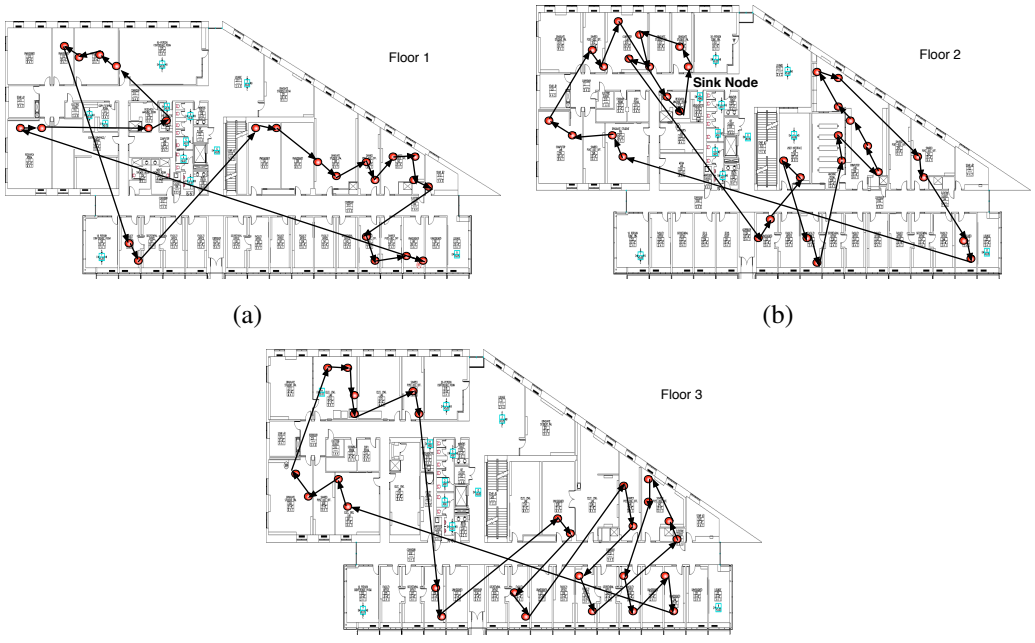


Figure 4.8 Most frequent activity sequences (order of active nodes) occurred in each floor of Motelab testbed during experiment

4.4.2 Evaluation in *Motelab* Tested

Evaluation environment: We have evaluated our proposed *EAR* protocol in large scale 82 node network of $TelosB$ motes (physically distributed in three floors, as shown in Figure 4.8(a), 4.8(b) and 4.8(c)) in Harvard *Motelab* sensor network testbed [58]. The experiments are conducted in 82 node network physically distributed across three floors.

Activity transition and data generation: From a separately deployed motion sensor network testbed we have learned the activity transition patterns and have validated the construction of activity transition graph $ATPG$. The activity transition patterns are modified to be scalable for a 82 node Motelab testbed, and is injected in the testbed for activity event generation and activity transition. The activity transition decides the order with which nodes will be active.

The activity event generation makes node(s) active, letting it send data to sink node (base station) at a high rate (we used data sending rate of 480 Bytes/second). We have emulated the activity events by generating three independent sequences of active nodes (indicating motion trails) each in one of the three floors. From a remote server, periodic serial message (containing new active node numbers) is sent to the sensor motes in the testbed to generate the activity sequences. The sensor nodes receiving the serial message with it's ID start generating sensor data. Other nodes act as relay only. This periodic activation of nodes through serial message follow the activity transitions defined in the corresponding $ATPG$. In this way the activity transition experiments are performed with networkwide data collection. In addition each node periodically sends one local status data packet (containing information of remaining energy, hop count etc.) to sink every 30 seconds.

Comparison: For performance comparison we have compared *EAR* with standard existing routing schemes to show performance improvement. Following relevant routing protocols are used: $PMin$ (shortest path routing), CTP [59] (very commonly used data collection protocol for sensor networks, that uses link and path quality), and $CMAX$ (an energy aware protocol [34] where data is forwarded preferably to neighbor with higher remaining energy in the neighborhood).

The 82 node network formed a 9 hop routing tree with -5 dBm transmission power of CC2420

radio of TelosB motes. The sink node is in middle of the three floors. In this network data collection scenario we have evaluated following parameters: (i) data delivery latency, (ii) data throughput, (iii) minimum node energy in the network through time (indicating network lifetime). Now we describe the performance analysis of our proposed *EAR* protocol compared to existing protocols.

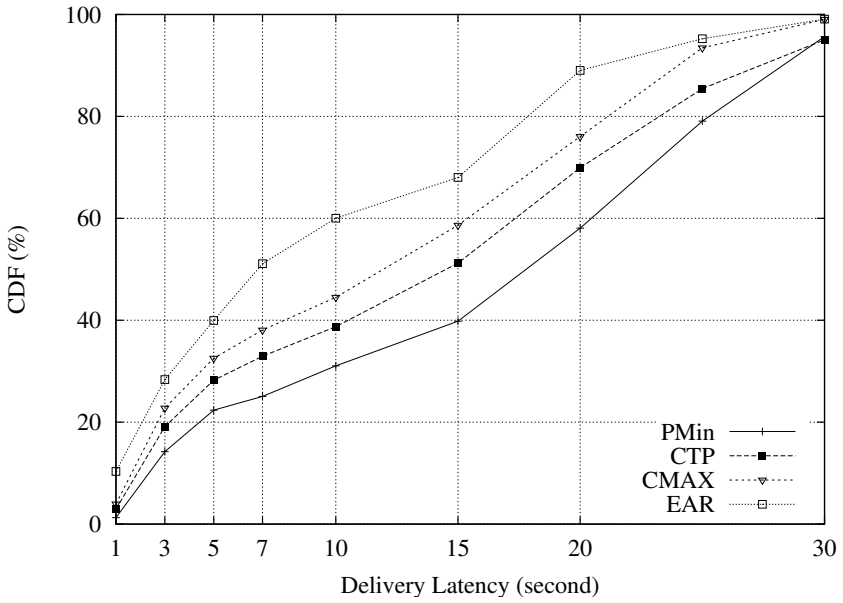


Figure 4.9 Distribution of data delivery latency.

Latency: Figure 4.9 represents the distribution of data delivery latency of packets in the 82 node network. It can be observed that *EAR* provides much lower latency than each of the comparing protocols *PMin*, *CTP* and *CMAX*. In *PMin*, *CTP*, *CMAX*, 80% of the packets are delivered with latency between 22 seconds to 25 seconds. But in *EAR* the 80% of the packets are delivered within latency around 18 seconds. Therefore *EAR* provides much lower delivery latency, providing better performance to the application. *EAR* achieves this improvement in latency by avoiding selection of currently active nodes (which are busy with sensing and sending own data) as relays.

Data Throughput: Figure 4.10 shows the data throughput for each node at sink. More throughput indicates more event data successfully delivered and reported at sink. It is observed that for each of the 82 nodes, *EAR* provides much improved data throughput than others. For all the 82 nodes *EAR* provides a data throughput improvement ranging from 6% to 13%. This

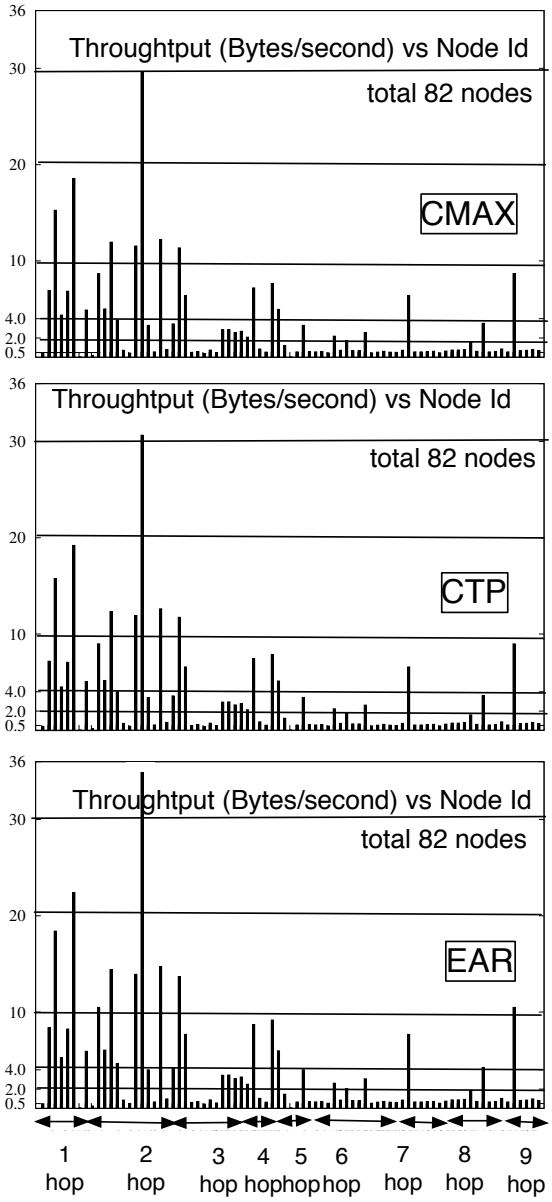


Figure 4.10 Data throughput at Sink.

advantage in *EAR* comes from avoiding selection of currently active nodes (which are busy with sensing and sending own data) as relays.

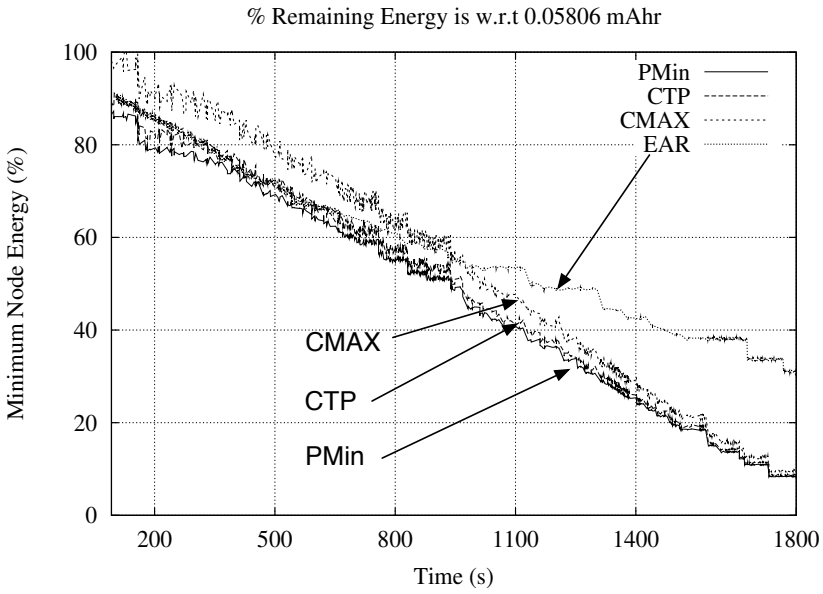


Figure 4.11 Minimum node energy in network.

Lifetime: Figure 4.11 represents the minimum energy of any node in the entire network through time. This property decides the network lifetime. The energy consumption is configured in such a way that all nodes start with energy 2200 mAh. For indicating the rate of drop in minimum remaining node energy in network, Figure 4.11 represents the energy drop with respect to a chosen value of 0.05806 mAh. This value is chosen for purpose of analysis because the minimum node energy in *PMin* reduces by an amount of nearly 0.05806 mAh energy in experiment run time of 1800 seconds (i.e. 30 minutes). This chosen value for analysis doesn't affect the nature of energy consumption of network. Now it can be observed that the minimum energy of any node in *PMin*, *CTP* and *CMAX* depletes faster than the one in *EAR*. Therefore in protocols other than *EAR* network node depletes almost all it's energy within time 1800 seconds, had the network start with 0.05806 mAh for all nodes. But in same scenario in same time the minimum energy of any node in *EAR* would have still around 32% energy left. Therefore it is clear that network lifetime for *EAR* will also be much higher than others. *EAR* achieves this advantage because of energy balanced

relay node selection.

4.4.3 Evaluation in TOSSIM Simulator

To validate the scalability of *EAR* we have used network size containing 20, 40, 60, 80 and 100 nodes (all with lossy wireless channel). The topology of 20 node network closely follows the node distribution in kitchen, dining room and bedroom of *CASAS* testbed, as shown in Figure 4.5. The topology of nodes in the other networks also follow the layout in Figure 4.5, but modified according to the network size. The activities are probabilistic and follow the activity transition patterns. We have generated *ATPG* with similar activity patterns for larger networks containing 40, 60, 80 and 100 nodes. When activities occur in a node, it performs some processing and then sends out bunch of data packets containing activity detection data.

The energy consumption is calculated using the relevant model of: CC2420 radio parameters (19.7 mA current consumption in receive mode, 17.4 mA current consumption in transmit mode, 250 kbps data rate with 48 kByte data packet size), and the MSP430 MCU parameters (3 mA current consumption in active mode due to sensing and computation). To note that due to timer and ADC read operations, sensor nodes can consume as high as 3mA current (as observed in [52]). The remaining node energy is updated accordingly.

EAR is compared with following relevant routing protocols: *PMin*, *CTP*, *MaxEn* (data forwarded to node with maximum remaining energy among the relay nodes) and *CMAX*. Each experiment with a network size is conducted for 2 hours. This generates multiple possibility paths of activity due to probabilistic activity transition in *ATPG*.

Data delivery latency: Despite preferring activity-aware and energy-balanced path, *EAR* also provides better data delivery latency. This is because of the activity aware property of *EAR*, which prefers less active nodes as forwarder (i.e. relay) node. This is validated from experimental results in Figure 4.12. For different network sizes, *EAR* provides from 6.8% to 19.1% less data delivery latency over others. *CMAX* and *MaxEn* are only energy-aware, so in non-uniform data generating network (leading to non-uniform energy nodes) the routed data packets sometimes deviate and follow a longer path. This leads to high data delivery latency. *PMin* has better data delivery

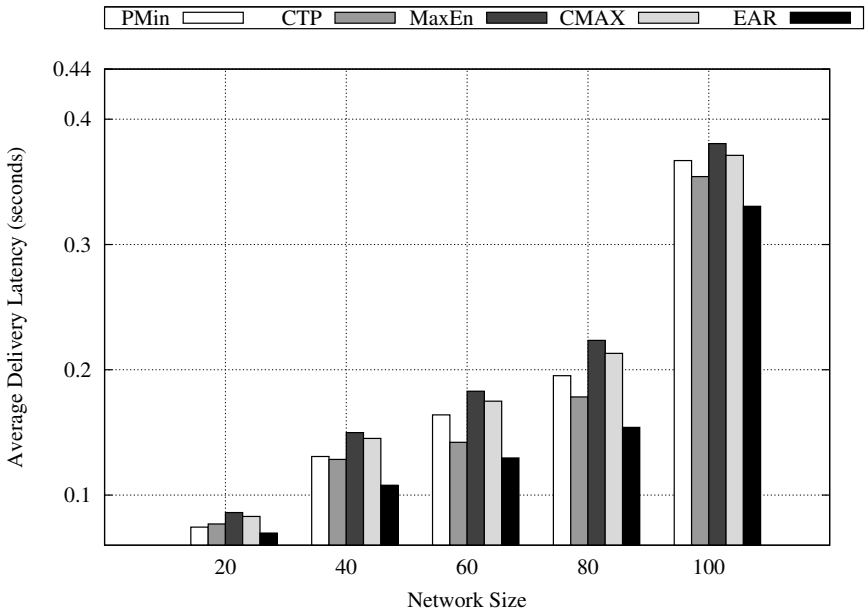


Figure 4.12 Mean data delivery latency (seconds) with varying network size.

latency by following shorter path, but suffers from retransmissions and from processing delay when being forwarded through active nodes (busy in sensing, processing and sending own data). *CTP* provides better delivery latency, but still performs worse than *EAR* because it doesn't learn from activity patterns.

Data throughput: *EAR* tries to minimize the network lifetime, with maintenance of network energy balance. Despite providing these advantages, *EAR* doesn't degrade the throughput (successful message received at sink per unit time) much. This is validated through results in Figure 4.13. *PMin* and *CTP* provide better throughput. But throughput performance of *EAR* closely follow (within upto 2% lesser) that of *PMin* and *CTP*. Energy-aware only protocols *CMAX* and *MaxEn* suffer worse throughput for lack of activity-awareness and lack of faster convergence in non-uniform activity generation network. To note that despite following activity and energy awareness, *EAR* make sure faster convergence by using hop spanner property discussed earlier.

Network lifetime: From experimental results in Figure 4.14 it can be observed that *EAR* achieves the maximum network lifetime for all the network sizes. For different network sizes, *EAR* achieves an improvement in lifetime over others from 9.31% to 23.77%. Figure 4.14 *CMAX* and *MaxEn* have better network lifetime than *CTP* and *PMin*, because *CMAX* and *MaxEn* are

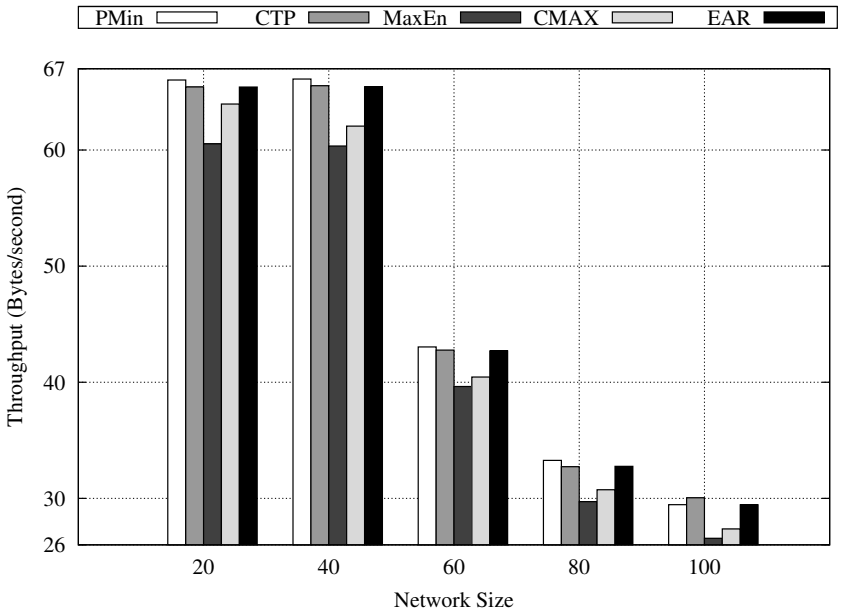


Figure 4.13 Data throughput at base station (successfully delivered message per unit time) in Bytes/second with varying network size.

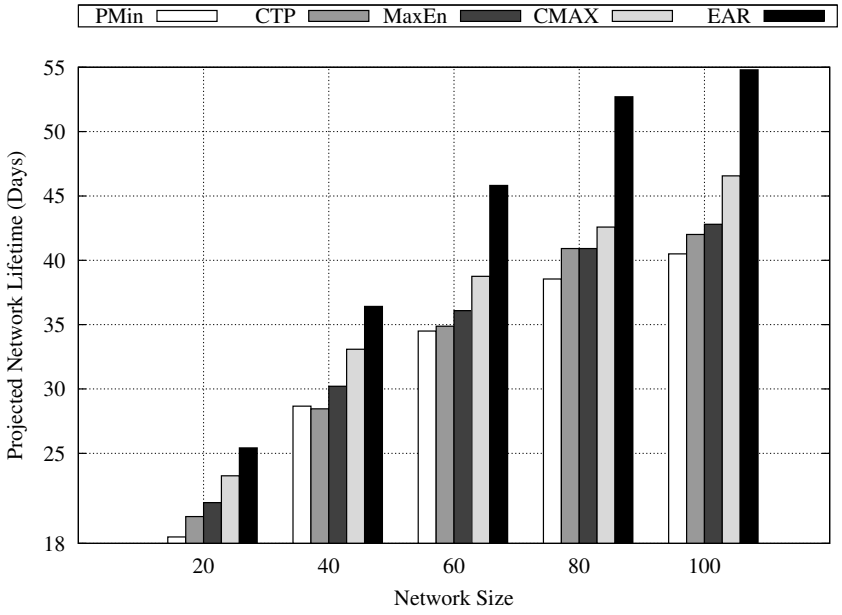


Figure 4.14 Projected network lifetime (days) with varying network size.

energy aware protocols. But their performance is worse than *EAR* due to inability to keep energy balance and to be activity-aware, *CTP* and *PMin* suffer because they are not activity-aware or energy balancing. This is interesting observation for networks where data generation is a non-uniform and dynamic process, but have some patterns.

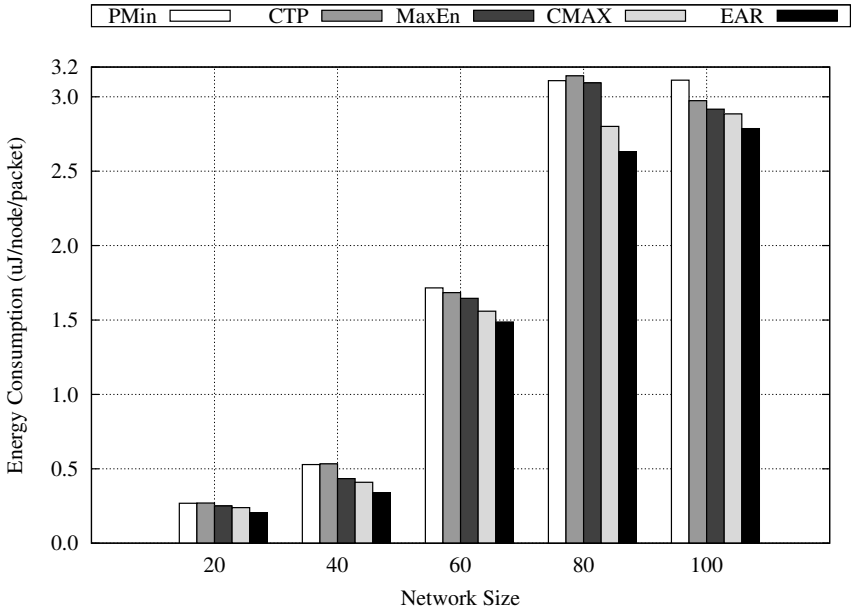


Figure 4.15 Energy consumption per successfully delivered message per node (uJ/packet/node) with varying network size

Network energy consumption: *EAR* also has minimum network energy consumption for all network sizes. In Figure 4.15 we have used the parameter for indicating effective network energy consumption. This is represented by the parameter: the total energy consumption per delivered packet per node. This indicates the average amount of energy spent by a node to enable one successful routing and collection of a data packet from network to sink. It can be observed that *EAR* has the minimum observed network energy consumption. For different network sizes, *EAR* provides from 3.4% to 17.2% improvement in network energy consumption over others. This proves the effectiveness of activity-awareness and energy balance of *EAR*. Due to energy balance property and avoiding active nodes for forwarding, the network as a whole spends less amount of energy for delivering data packets.

Scalability: All the advantages of *EAR* are achieved for network size varying from 20 to 100.

This proves the scalability, thus its real-world applicability for pervasive environments.

4.5 Summary

In this chapter we have presented our proposed *EAR* for activity-aware and energy-balanced routing. As a case study *EAR* is evaluated with Smart Environment data trace. The experimental results have demonstrated its efficiency both with respect to application and network performance, as well as its scalability.

PART 5

ACTISEN: ACTIVITY-AWARE SENSOR NETWORK SYSTEM

In this chapter we present the *ActiSen* [3] system, Activity-Aware Wireless Sensor Networks for Smart Environments. First we describe the sensing, radio duty cycling and routing protocols in *ActiSen* in details. Then we present the system evaluation and performance analyses of *ActiSen*.

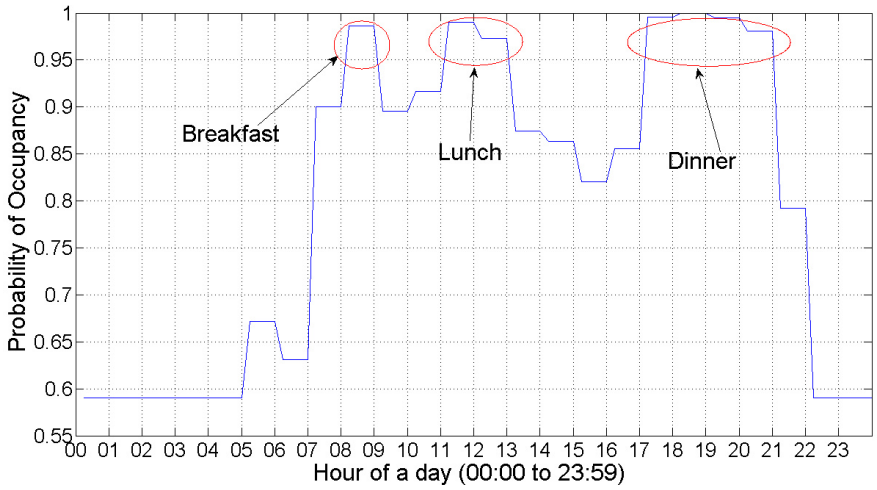


Figure 5.1 Probability of occupancy in the kitchen of a smart home for assisted living (CASAS [5]), detected by motion sensor.

5.1 Background

Wireless sensor networks have enabled many important social and scientific applications and its protocol design has received considerable research interest. But many existing works did not realize an important difference between sensor networks and traditional networks. Unlike a traditional communication network, a sensor network is deeply embedded in environments and its operation is driven by the activities in the environment. In many applications such as Smart Environments, the information of event activity shows certain patterns in long run (as shown in Figure 5.1). Most of the sensor network designs till date under-utilized the activity pattern for

performance improvement of the network. To note that in this paper activity is meant by the events that are sensed and reported by the nodes in sensor network. For example in a Smart Home environment, one type of activity is the motion activity of the residents. There remains a missing link in sensor network design: *the feedback from sensed and analyzed activity pattern, back to the network operation for resource usage*. The activity pattern information, if utilized in an intelligent manner, can improve the sensor network performance while reducing resource usages.

5.2 *ActiSen*: Activity-Aware Sensor Network System

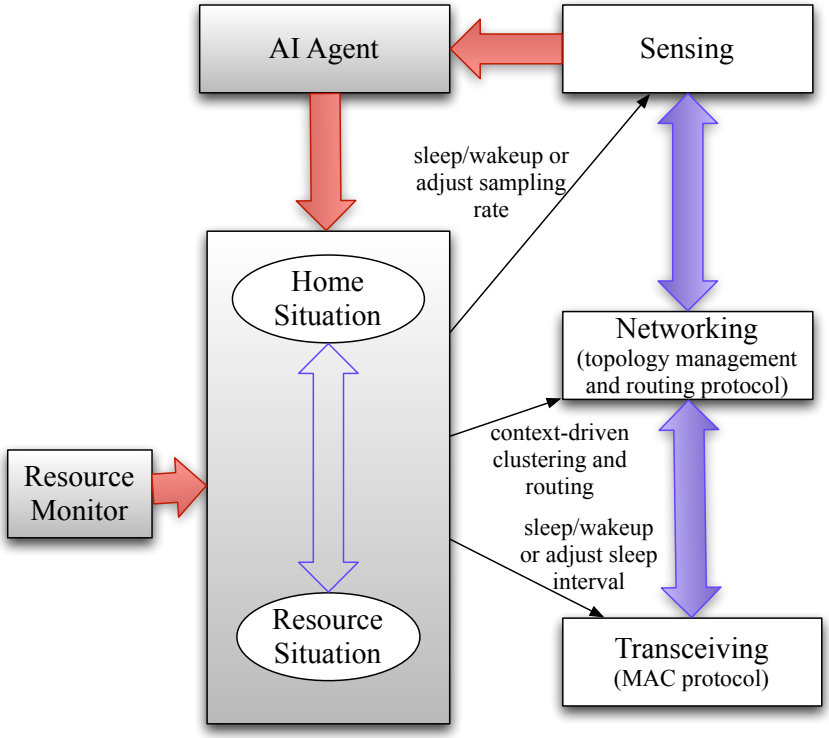


Figure 5.2 The system architecture of *ActiSen*

ActiSen adapts sensing, radio duty-cycling and routing according to its historical activity pattern or transition information. Figure 5.2 presents the whole system architecture of *ActiSen*. An AI agent is trained from the regular activities and situations in the application environment. The agent provides *Activity Transition Probability Graph* (*ATPG*) (an example shown in Figure 5.13),

which contains information about transition probability and transition duration of all the activities, predicted in the context of time and space. Figure 5.13 shows the $ATPG$ with 27 nodes, learnt from the CASAS Smart Home testbed ([5]). Using such intelligence about predicted activities, the sensing, radio duty-cycling, and routing are dynamically configured for optimized operations. The design of *ActiSen* includes:

- An activity-aware sensing scheme that achieves high event detection accuracy while reducing energy consumption through adaptive sampling intervals.
- An activity-aware radio duty-cycling protocol that dynamically adapts the radio's duty-cycle for low latency delivery, while maintaining high energy efficiency.
- An activity-aware and energy balanced routing protocol, that jointly considers activity patterns and residual network energy to balance energy consumption rate across the network thus prolonging network lifetime.

Now we describe the essence of *Activity Transition Probability Graph* ($ATPG$). In a Smart Home at certain time of the day, a resident's Activities of Daily Living (ADL) (say activity A_i) in a region is monitored by a set or cluster of sensors, say C_i. It is worth noting that *ActiSen* system and its protocols are designed for not only single source of activity (e.g. single resident), but generally for multiple activities (e.g. multiple smart home residents). Then in the smart home scenario, after some time the resident can probabilistically move to either cluster C_j or C_k or some other cluster. The activity-aware sensor network in such scenario can intelligently utilize its resource using the knowledge from an Activity Transition Probability Graph or $ATPG$ (say G_{act}). Each node of G_{act} denotes some specific cluster C_i. The edge from node C_i to node C_j denotes the transition tuple $< p_{ij}, t_{ij} >$ from C_i to C_j. p_{ij} is the predicted transition probability and t_{ij} is the predicted transition time. In *ActiSen*, the graph G_{act} is learned and updated by an AI agent.

After learning the ATPG G_{act}, the activity-aware sensor network utilizes it to optimize its sensing, radio duty-cycling and routing operations. The protocols in *ActiSen* are designed in such a way that activity detection sensors and the wireless radio can sleep as much as possible, while

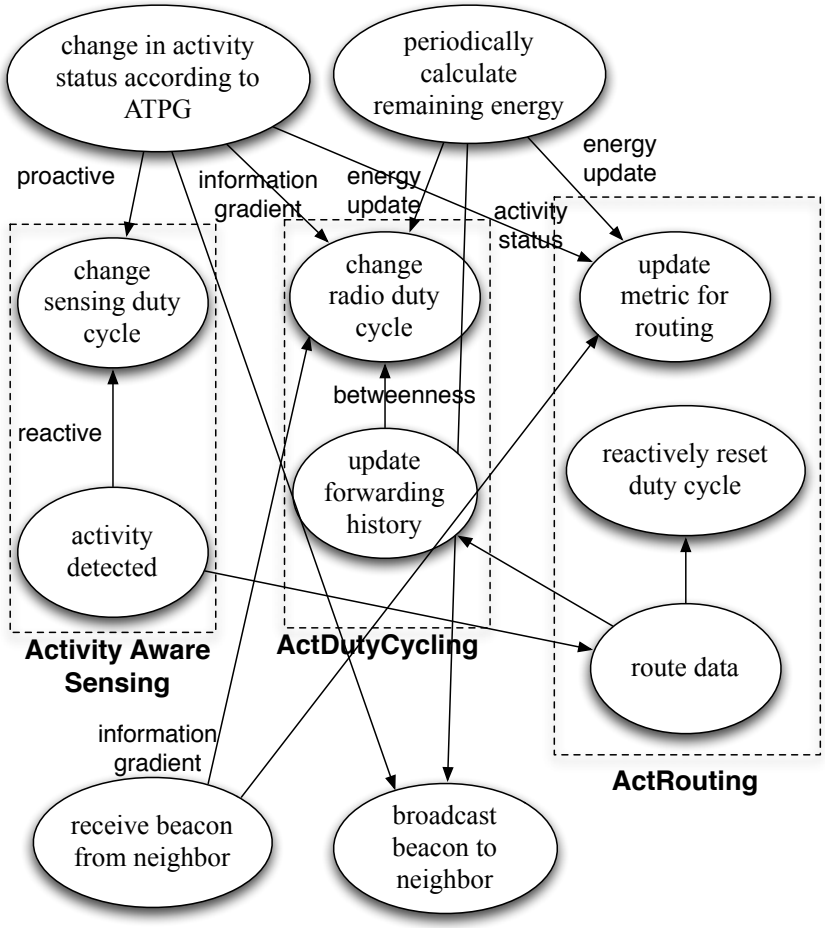


Figure 5.3 Workflow diagram of *ActiS en* system and inter-relationship among (a) Activity-Aware Sensing, (b) *ActDutyCycling*, and (c) *ActRouting*

the activity events are reliably covered, detected and reported to the sink (i.e. the base station). Now we present an overall description of how the network in *ActiSen* works for the example of smart home scenario. All the activity-aware protocols in *ActiSen* are later described in detail.

The design of sensing, radio duty-cycling, routing and their inter-relationship in *ActiSen* system is shown in Figure 5.3. Each of the algorithms are later described in details in next subsections. Here is a brief description of computation overhead of different components (sensor nodes and sink). (i) The event detection data in network is collected in sink node for application purpose, and used for constructing ATPG activity pattern. The activity pattern analysis is only done in sink. The activity transition pattern is disseminated into the network only once and updated if any change happens in pattern, which is very rare. (ii) All other calculations involved in *ActiSen* are distributed and localized in the network. The sensor duty cycle calculation uses locally stored activity information. *ActDutyCycling* uses information of own and neighbors to calculate local node radio duty cycle. *ActRouting* also uses own and neighbors information only for deciding next hop node relay node. So all computations, except activity pattern analysis are distributed and localized in the network. This makes it practically applicable to networks independent of size.

Here is a brief description of communication overhead of different components (sensor nodes and sink). (i) All the sensor nodes in the network continuously sense activity and send the activity data through multi-hop network data collection to the sink node. This is the convergecast or data collection communication that goes on continuously whenever activity event is triggered. (ii) The sensor nodes, in their local 1-hop neighborhood, periodically exchange (through 1-hop broadcast communication) beacon message and share node or local neighbor information. (iii) Each node also periodically sends (network-wide data collection communication) one local status data packet (containing information of remaining energy, duty cycle, hop count etc.) to sink node with a relatively large period. (iv) If there is any major change in the analyzed activity transition pattern in the sink, the new activity transition information is disseminated into the network from the sink node (through data dissemination communication). In *ActiSen* system it is not needed frequently to disseminate data into network from sink. The dissemination can use any standard data disseminating protocol (e.g. *Cascades* [60]).

From the activity transition probability graph G_{act} for current activity, a cluster C_i of sensor nodes is constructed. The selection of member nodes of C_i is determined by the location context of activity. Then the nodes which are not member of C_i can turn their activity detection sensors ON less frequently. The nodes in C_i turn their activity detection sensors ON more frequently. Example of cluster is kitchen cluster where resident activities can occur for considerable time at certain phases of ADL (Activities of Daily Living). Now according to the activity transition graph, the active cluster is changed from C_i to next active cluster C_j, for the next predicted activity context. After the predicted transition time t_{ij}, C_i triggers C_j to wake up and reconfigure. The previous cluster C_i can be kept ON for a marginal time to watch for any remaining activity. Now we explain in details, how sensing, radio duty cycling and routing are performed in *ActiSen* system.

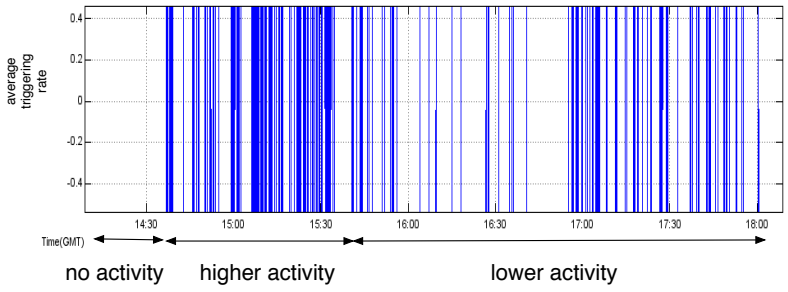


Figure 5.4 Changing rate of activity detected; PIR (passive infrared) motion sensor data (sampling frequency 10 Hz) shown with transition from no activity to higher activity and then to lower activity.

5.3 Activity-Aware Sensing

Figure 5.4 shows an example of detected PIR motion sensor signal for a typical transition of human activity among different intensities. Maintaining a fixed sampling interval for sensing could result in either high energy consumption (when interval is small) or low detection accuracy (when interval is big). Motivated by the need of an activity-aware adaptive sensing scheme, we have proposed the following algorithm. Each node in the network proactively and reactively adapts the sampling intervals.

Algorithm 4 *ActSensing*: Activity-aware Adaptive Sensing

Each node i dynamically adjusts sampling intervals as follows:

loop
  if $i \in (Sleep|Quasi-Active|Active)$ **then**
    $T_{OFF} = (T^s_{OFF}|T^q_{OFF}|T^a_{OFF})$
  while *no source transition* **do**
    if $T^{peak}_{min}(t) < T^{peak}_{min}(t-1)$ **then**
      $DC(t+1) = (DC(t) + \delta)$
    else
      if $T^{peak}_{min}(t) > T^{peak}_{min}(t-1)$ **then**
        $DC(t+1) = (DC(t) - \delta)$
      else
        $DC(t+1) = DC(t)$

Proactive selection of base sampling interval: As described in Section 5.2, each sensor node belongs to one of the three sets regarding role in activity sensing: *Sleep* (nodes outside active or next predicted clusters), *Quasi-Active* (nodes inside the next predicted clusters and outside active clusters), *Active* (nodes inside the active clusters). Then the sampling interval for activity sensing is dynamically configured based on the type or role of the node at current moment. The activity detection sensor is ON for T_{ON} and is then OFF for time T^s_{OFF} (for $Sleep$) or T^q_{OFF} (for $Quasi-Active$) or T^a_{OFF} (for *Active*), such that $T^s_{OFF} > T^q_{OFF} > T^a_{OFF}$. This technique lets only the nodes in active regions sense more frequently, and the other nodes sense less frequently, thus saving energy. In addition to wake and sleep schedule for sensor, the nodes keep two more timing information. Once the sensor is ON, it only samples it after a stabilization delay T_{STAB} to remove initial erroneous samples. Also the sensor is kept ON for an elastic margin time for allowing certain fixed N_{SAMPLE} number of samples in case a motion is detected inside T_{ON}. N_{SAMPLE} samples are needed by application for detecting level of activities. Thus adaptive sampling interval is needed for the purpose of capturing important motion activity.

Reactive adjustment of sampling interval: In addition to proactively configured base sampling interval for sensing, the proposed design lets the individual nodes reactively adjust the sensing frequency in a finer scale with intensity of activity. When the activity frequency is low, a higher sampling interval is fine for detecting the events. But when activity frequency goes high,

the sensing interval can be adaptively decreased to enable successful event detection. This allows the nodes to potentially save more energy in idle situation, but also remain adaptive to changing activity level. Let $T_{min}^{peak}(t)$ is the minimum timegap between two consecutive peaks in detected activity signal during t-th interval of sensing. Then from tracking $T_{min}^{peak}(t)$ during intervals, the sensing frequency $DC(t+1)$ is further adjusted with a finer scale of say δ. For example, the scale δ is 1%. The reactive adjustment of sampling interval is also useful near the transition of data source. Overall the proposed activity-aware sensing algorithm proactively and reactively adapts the sensing frequency for saving energy, and assuring reliable event detection. The $ActSensing$ is formally described in Algorithm 4.

Activity event beyond prediction: The activity-aware sensing algorithm is protected from missed detection of unusual activity that don't follow the predictions in $ATPG$. T_{OFF}^{s} can be set to a low but safe value based on human motion frequency. Then on occurrence of such unusual event, activity-aware sensing algorithm will be able to detect it. Once detected, the reactive adjustment property will increase the sampling frequency for finer activity detection. Alternatively *sensor wake-on* hardware property (in [52]) can be used by low energy cost sensor (always on) to wake up higher energy cost sensor to detect activities. But this alternative can be applied in sensing only some physical phenomena.

5.4 Activity-Aware Radio Duty-Cycling

In this section we describe our proposed activity-aware radio duty-cycling protocol. In a typical wireless networks, a one-hop packet delivery latency usually includes processing delay, transmission delay, and propagation delay, which are usually in milliseconds order. However, in a low-duty cycle network, a sender may need to wait for its receiver to wake up before it can send a packet. Thus sleep interval dominates the overall delivery latency. Therefore, in this work we only consider sleep latency as notion of delay.

Need of non-uniform radio duty-cycling:

Scheduling the operation of the wireless radio is a crucial task in achieving efficient perfor-

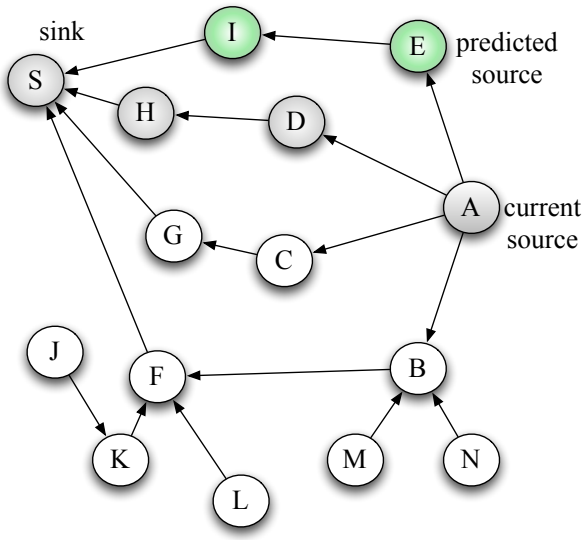


Figure 5.5 Scenario showing need of activity-aware radio duty-cycling

mance for wireless sensor networks. Now in the network at any time only certain active region of sensor nodes generate data. This scenario is shown by an example in Figure 5.5. Suppose in the network topology, the currently active data source is node *A* and predicted active data source is node *E*. To note that in general case there can be multiple current or predicted active (data generating) nodes in network. Let the probable active routing path is *A*-*D*-*H*-*S*, and the probable routing path for predicted source is *E*-*I*-*S*. Then an efficient radio duty-cycling policy is: (i) to maintain high duty cycle for nodes on and near the routing paths for active and predicted sources (for fast and reliable data delivery on any meaningful route selected), and (ii) to maintain low duty cycle (for energy saving) for potentially idle nodes away from active routing paths (for example nodes *J*, *K*, *L*, *M*, *N* etc.). The major challenge for maintaining such non-uniform duty cycling is that, the distribution of duty cycle has to dynamically adapt with change of active and predicted sources. In this aspect we have designed *ActDutyCycling*, an activity-aware radio duty-cycling protocol. It is not just more energy efficient but also provides reduced latency for delivery of data to sink. The potential of *ActDutyCycling* is that it can dynamically adapt the radio duty-cycling operation across the network, in the context of current and predicted active regions.

Before describing *ActDutyCycling* in detail, we define the parameters pertaining to the radio duty-cycling decision. The parameters for each node (say i) are described as follows:

Information Gradient (g_i): Information potential or information gradient g_i (as described in [61] and [62]) is a real valued function for each node i. It is defined as a function that meets the following requirements: its value is (a) 0 at the sink, (b) 1 at nodes on the active/predicted cluster, (c) at any other node, the function value equals the average value of its neighbors. It can be shown that there is a unique such function for any given source (it is the harmonic function meeting the specified boundary conditions). The information gradient is a 'smooth' function with no local extrema. It is stable to small changes in network connectivity. Effectively the information gradient g_i indicates the estimate of the nodes' relative position between active/predicted cluster (source of data) and the base station (sink of data). This parameter is used in *ActDutyCycling* for controlling the radio duty cycle in context of activity and node's possible role as relay. *ActDutyCycling* actually uses two gradient values: (a) information gradient for current data source (gc_i), and (b) information gradient for predicted data source (gp_i).

Hopcount (c_i): It is a representative of the depth or the minimum hopcount from node i to the base station. For scaling to maximum value of 1, c_i is the ratio of hopcount to base station (say hop_i) to the estimated depth of the total network tree (say D). Then, $c_i = hop_i/D$. For practical issues, if the depth of the network is hard to maintain, then one of the two alternative definitions can be used. (i) c_i can be considered as the ratio of hopcount to base station to the total number of nodes in network (say N). Then, $c_i = hop_i/N$. (ii) If ideally c_i has to be a local property of node, then it can be defined as $c_i = (1\text{-}1/hop_i)$. In experiments we have used the first definition of c_i.

Betweenness(b_i): Based on the activity transition probability graph and activity-aware routing path designed for each predicted source, it is easy to calculate the probability that a given sensor i will be involved in the routing path after the next source transition. The intuition behind our design is that the sensors that have higher probability of becoming relay nodes in the next source transition, deserve higher duty-cycle. We introduce a concept, betweenness, that have been widely used in graph theory to describe the importance of a given vertex. Basically, vertices that occur

on many shortest paths between other vertices have higher betweenness than those that do not. In this work, we redefine the betweenness of a vertex as the probability that it will appear in the activity-aware routing path on the next source transition. Assume m is current source, $p_{m,j}$ is the transition probability from m to j. Let $I_j(i)$ indicate whether i will appear at the routing path from j to base station, e.g. $I_j(i) = 1$ if i will appear in the routing path and $I_j(i) = 0$ otherwise. $I_j(i)$ can be obtained immediately after the activity-aware routing path is found (as discussed in next section). Thus, the betweenness of given sensor i can be formally defined as: $b_i = \sum_{j \in V} p_{m,j} \cdot I_j(i)$.

Relative remaining Energy (e_i): It is an indicator of remaining energy of node i with respect to the nodes in its neighborhood. For scaling to maximum value of 1, e_i is represented as the ratio of remaining energy of node i to the maximum of the energy of the neighbors and itself.

Now we describe our proposed activity-aware radio duty-cycling protocol, we call *ActDutyCycling*. According to the data sending rate of the application, the radio on each node has a maximum sleep interval (say TS^{max}) and a minimum sleep interval (say TS^{min}). The naive radio duty-cycling will select a static and uniform sleep interval TS^{naive} for each node in the network ($TS^{min} \leq TS^{naive} \leq TS^{max}$). The radio ON time will be controlled by MAC protocol selected.

But *ActDutyCycling* prefers non-uniform duty cycle, and assigns duty cycle to nodes based on their expected role in data delivery and available energy. In *ActDutyCycling*, a fraction of TS^{max} is dynamically assigned as the sleep interval TS^{acti} (still satisfying $TS^{min} \leq TS^{acti} \leq TS^{max}$). This selection of TS^{acti} is updated with time, is non-uniform across nodes, and is a function of the parameters (described above): information gradient for current source, information gradient for predicted source, hopcount, betweenness and relative remaining energy.

Determination of Sleep Interval: Suppose cluster C_i is active at current moment and cluster C_j is predicted to be the next active cluster. Then *ActDutyCycling* attempts to control the radio duty cycle of the sensor nodes in the network, in order to satisfy following requirements:

(a) Reliable and low latency delivery of sensed data from active cluster node to the base station.

(b) Assurance of future activity detection accuracy, by setting up active path in advance, from predicted cluster to the base station.

(c) Low duty cycle for nodes in inactive region for energy saving.

The effective sleep interval $TS_i^{acti}(t)$ for node i is a function f($gc_i(t)$, $gp_i(t)$, $c_i(t)$, $b_i(t)$, $e_i(t)$) such that:

(a) As information gradient gc_i or gp_i increases, TS_i^{acti} decreases. Information gradient is highest (fixed at 1) at nodes in the active and predicted clusters, and is lowest (fixed at 0) at the base station. So the radio of the nodes near active/predicted cluster are made to be active more for data relay.

(b) As hopcount c_i to the base station increases, TS_i^{acti} increases. So radio of the nodes near base station are made to be active more for data relay.

(c) As betweenness b_i of a node increases TS_i^{acti} decreases. So probable nodes on the routing path from next predicted source are made to be active more.

(d) If relative remaining energy e_i of node i is below a critically low threshold (say e_{low}), TS_i^{acti} increases. So radio of any node with critically low remaining energy in neighborhood can sleep more for saving energy, and try to participate less in any data delivery.

Algorithm 5 *ActDutyCycling*: Activity-aware Radio Duty-cycling

if $i \in (sink \cup active\ cluster \cup predicted\ cluster)$ **then**
  $TS_i^{acti}(t) = TS^{min}$
else
  if $e_i(t) \leq e_{low}$ **then**
    $TS_i^{acti}(t) = TS^{max}$
  else
    f=$min((1\text{-}gc_i(t)), (1\text{-}gp_i(t)), c_i(t), (1\text{-}b_i(t)))$
    $TS_i^{acti}(t) = max(TS^{max}.f, TS^{min})$

Now *ActDutyCycling* works as follows. Each sensor node i in the entire network keeps track of: hop count c_i, relative remaining energy e_i, betweenness b_i and two gradient values (gc_i for currently active cluster, gp_i for predicted cluster). The c_i, e_i, b_i, gc_i and gp_i are periodically updated

in a distributed manner among the neighbors. Then each node i computes the effective sleep interval $TS_i^{acti}(t)$ as shown in Algorithm 5.

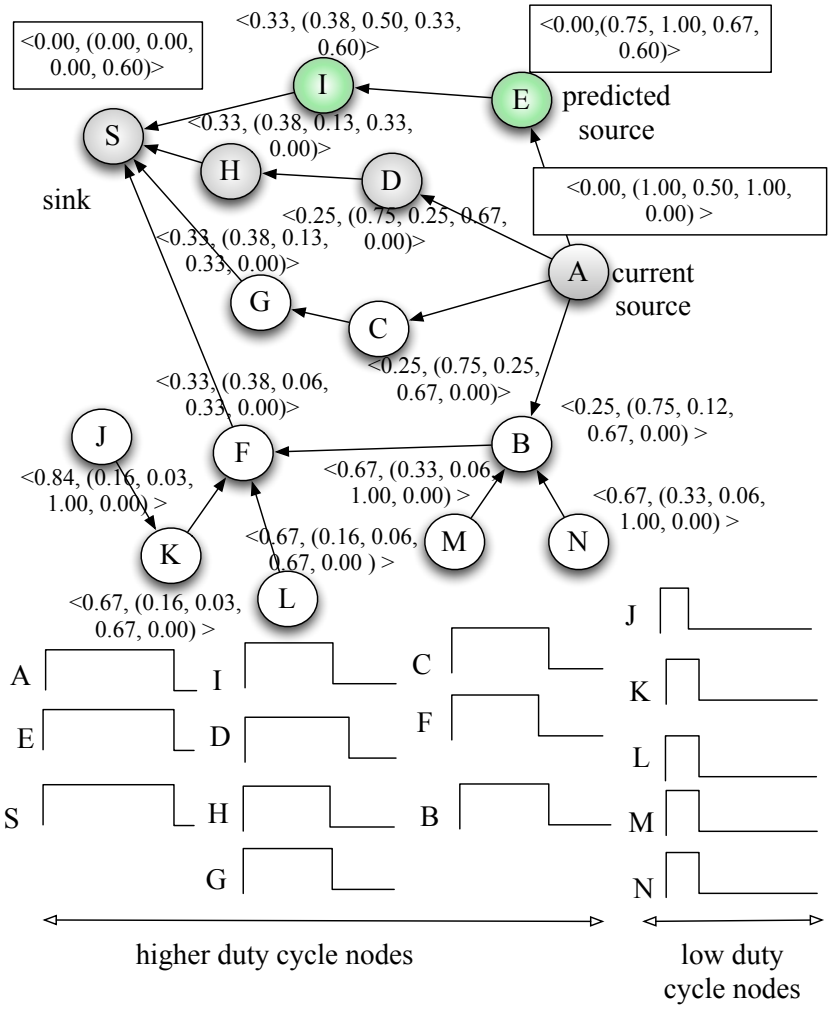


Figure 5.6 Non-uniform Duty cycle control in *ActDutyCycling*. The set of parameters < f, (gc, gp, c, b) > for each nodes are shown. It is assumed that the predicted source transition is from node A to node E with a probability 0.60

The working of *ActDutyCycling* is illustrated in Figure 5.6 in a network topology same as in Figure 5.5. At certain moment in the network, the parameters of *ActDutyCycling* for each node are shown. Then using gradients, hopcount, betweenness and energy, the nodes calculate their individual sleep interval. As shown in the figure: (i) sink (S), current source (A) and predicted source (E) select minimum sleep interval (i.e. maximum duty cycle), (ii) the nodes on or near

active route between current/predicted source and sink (nodes *I*, *H*, *D*, *G*, *C*, *B* and *F*) select relatively low sleep interval (i.e. higher duty cycle), (iii) nodes away from active region (nodes *J*, *K*, *L*, *M* and *N*) select larger sleep interval (i.e. low duty cycle). Here node *I* has a high betweenness, so selects relatively low sleep interval (i.e. higher duty cycle). Therefore in this activity context *ActDutyCycling* lets only meaningful nodes keep awake more, while letting other nodes sleep more.

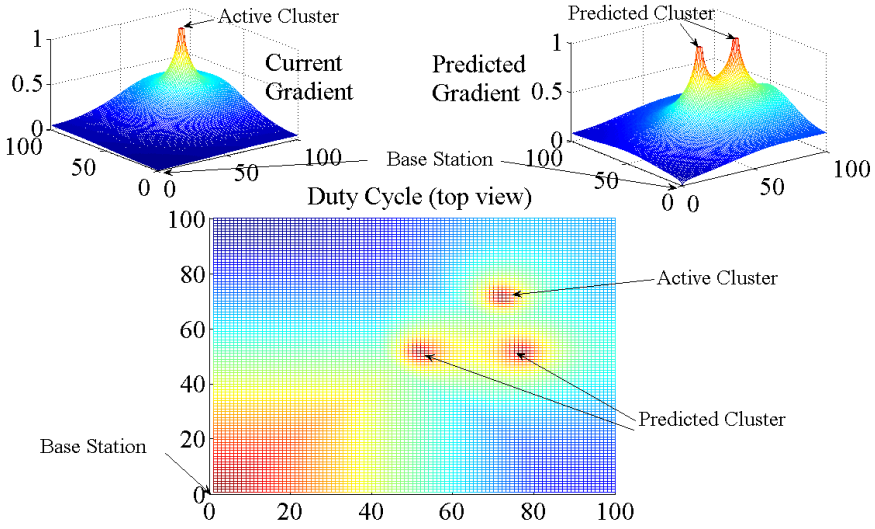


Figure 5.7 Activity-Aware non-uniform radio duty-cycling in network

The effectiveness of *ActDutyCycling* is also shown in a simulation environment of 100x100 sensor grid in Figure 5.7. As shown in Figure 5.7, the nodes in the grid maintain information gradients for currently active and predicted clusters. Then according to *ActDutyCycling* mechanism the nodes calculate their individual duty cycle. It can be clearly observed that only a set of nodes between base station and possible data source (active and predicted clusters) maintain high duty cycle, while the other nodes in network keep a relatively low duty cycle. This shows the advantage of using activity-aware radio duty-cycling, leading to energy efficiency and at the same time assuring reliable and low latency data delivery.

The role of *ActDutyCycling* in the whole *ActiS en* system and the relationship of *ActDutyCycling*

with sensing and routing components is illustrated in Figure 5.3.

5.5 Activity-Aware and Energy-balanced Routing

In *ActiSen*, at any moment, only some set of nodes perform activity detection. This non-uniform distribution of data source in network leads to non-uniform energy consumption of nodes in the network. In such scenario an efficient routing solution is the one with: (i) awareness of activity context and (ii) maintenance of energy balance across network. The nodes in active region can be excluded from task of data relay as much as possible. This will keep the processing unit (MCU) of the active nodes available for sensing, processing and communicating. Then there are other nodes in the network available who are not performing any activity detection. These nodes have their energy and MCU available for relaying the data stream to the sink. Therefore meeting network-wide energy balance and activity-awareness are also interlinked to some extent.

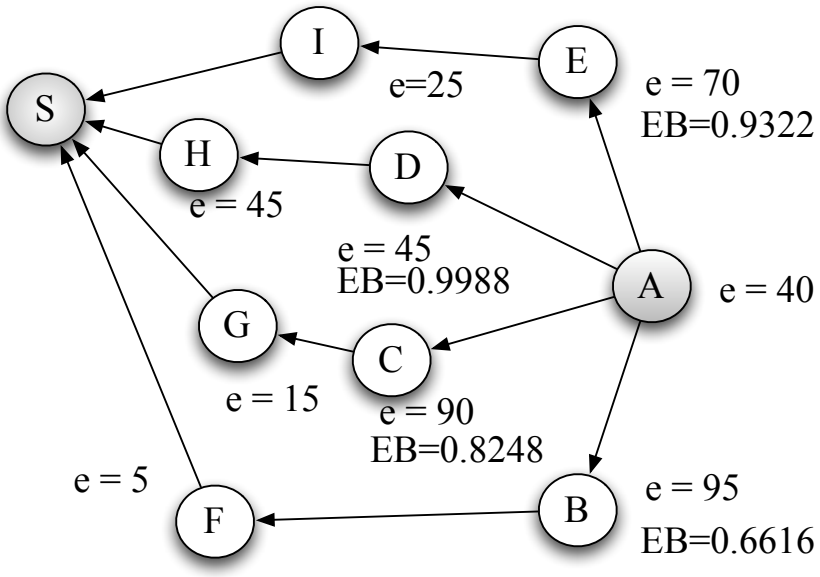


Figure 5.8 Energy balanced routing

Network energy balance is a critical issue in routing for achieving longer network lifetime and reduced network energy consumption. A greedy routing scheme will exploit nodes with good path quality or node with high energy, leading to energy drain of certain nodes and eventually network

failure. For example, in the network in Figure 5.8, nodes A, B, C, D, E, F, G, H and I have their remaining energy e as shown. Node A, the currently active data source, has to decide the relay node from B, C, D, E. An energy greedy algorithm, looking for maximum energy neighbor, will choose path A-B-F-S. Another energy greedy routing scheme, looking for path with maximum energy, will choose path A-C-G-S. But in all of these cases the node with low energy (F, G or I) will soon drain its energy, leading to network failure. But an energy balanced routing, with localized knowledge about degree of energy balance in network, is more efficient. Such energy balancing routing algorithm will choose a better energy balanced path A-D-H-S. This can increase the network lifetime considerably. An energy balanced routing is different from max-min routing. A max-min routing protocol selects a path with maximum of the minimum energy. But our energy balanced protocol prefers the local energy balance in the network for routing decision. This can be more useful also in case there is data flow from multiple sources. In that case for max-min, all the flows will be directed towards the max-min energy node. This will be detrimental to the cause resulting in increased routing stretch and increased energy consumption of some bottleneck nodes. Then instead of assigning locally energy balanced path for each flow will be more effective.

Through MATLAB simulation in large scale network, it has been confirmed that our proposed *ActRouting* performs much better than other energy aware routing techniques (maximum energy neighbor, maximum energy path, max-min energy). The simulation environment is set up as follows: in each time epoch each node in the network generates and sends a flow of packets and forwards received packets from children in last epoch. The base station is placed at one corner of the grid. The average lifetime of a number of experiment runs is shown in Figure 5.9. One more useful observation from large scale network simulation is that, *ActRouting* performs even better in the scenario where nodes boot up with uneven distribution of remaining energy. Therefore the proposed routing algorithm can efficiently extend the lifetime of a network even with considerable energy imbalance.

The proposed *ActRouting* is based on activity context and energy inequality. It employs intelligently energy balanced data delivery in local region of the network, leading to global energy balance and improved network lifetime. The rationale behind using routing based on local energy

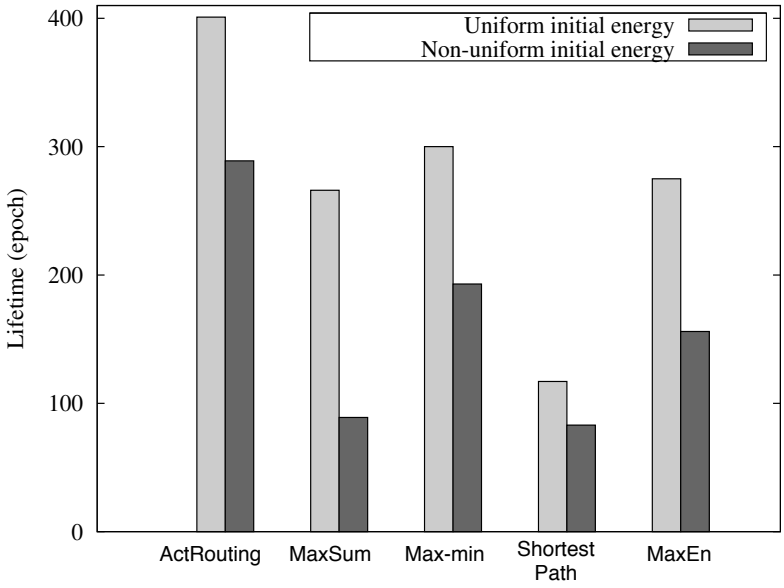


Figure 5.9 Network Lifetime of 30×30 grid network for different routing algorithms (MaxSum: maximum sum of energy of routing path, Max-Min: maximum of minimum energy on routing path, ShortestPath: shortest path (minimum hop) routing, MaxEn: maximum energy neighbor).

balance is that, it can guide the data through energy balanced path without requiring any global knowledge or large network state. *ActRouting* also handles the situation where local region is energy balanced, but all the nodes have low energy. Atkinson's Inequality Index [55] is used in *ActRouting* for indexing local energy balance, because it uses local entropy and the index can also be controlled based on needed degree of energy balance.

Atkinson Index: In order to reach a balance in energy consumption rate across the network we use Atkinson's Inequality Index [55]. It is a measure of economic income inequality in a society. The index can be turned into a normative measure by imposing a coefficient ε to weight incomes. Greater weight can be placed on changes in a given portion of the income distribution by choosing ε, the level of *Inequality Aversion*. The Atkinson index becomes more sensitive to changes at the lower end of the income distribution as ε approaches 1. Conversely, as the level of inequality aversion falls (ε approaching 0) the Atkinson Index becomes more sensitive to changes in the upper end of the income distribution. Atkinson index AT is defined as in equation 5.1.

$$AT = 1 - \frac{1}{\mu}(\frac{1}{N}\sum_{i=1}^{N} y_i^{(1-\varepsilon)})^{1/(1-\varepsilon)} \tag{5.1}$$

Where $0 \leq \varepsilon < 1$, y_i is the individual income of i-th entity (i = 1, 2, ..., N) and μ is the mean income of total N entities. We have used the Atkinson index AT for the scenario of sensor networks where the income parameter is replaced by the parameter normalized remaining energy $re_i(t)$, the remaining energy of node i at time t divided by the maximum energy capacity. Therefore Atkinson index AT of the entire network at time t is as shown in equation 5.2.

$$AT(t) = 1 - \frac{1}{re_{avg}(t)}(\frac{1}{N}\sum_{i=1}^{N} re_i(t)^{(1-\varepsilon)})^{1/(1-\varepsilon)} \tag{5.2}$$

Where $re_{avg}(t)$ is the average of the remaining energy $re_i(t)$ of all the nodes in the network at time t, N is the number of nodes in network. Since this index contains information of all the nodes in the network, only a centralized algorithm can compute the index completely. But it is possible to apply a distributed protocol where every node can compute the index locally within its 1-hop neighborhood. We apply the formulated Atkinson index AT in the problem of data forwarding. When a node has data packet to forward, the research challenge is to choose the relay node from the next available neighbors. The use of Atkinson index AT penalizes large inequality in remaining energy. An energy balanced relay node selection tries to maintain a balance in energy consumption rate of the nodes in the local region. This local balance in turn results in balance in energy consumption across the whole network.

Energy Balance Metric: Now we formulate our proposed routing metric, called Energy Balance (say EB) for relay selection. We design EB as (1-AT_{local}), where AT_{local} is the Atkinson Index computed locally in 1-hop neighborhood. Since AT_{local} denotes the level of energy inequality in 1-hop neighborhood, EB here denotes the level of energy equality or energy balance in 1-hop neighborhood. So at time t the routing metric $EB_i(t)$ computed by each node i is as shown in equation 5.3.

$$EB_i(t) = \frac{1}{re_{avg}(t)}\left(\frac{1}{|\Delta|}\sum_{i=1}^{\Delta} re_i(t)^{(1-\varepsilon)}\right)^{1/(1-\varepsilon)} \tag{5.3}$$

To note that for each node, Δ is the set of 1-hop neighbors and the node itself. So the metric EB is calculated using remaining energy information of the neighbors and the node itself.

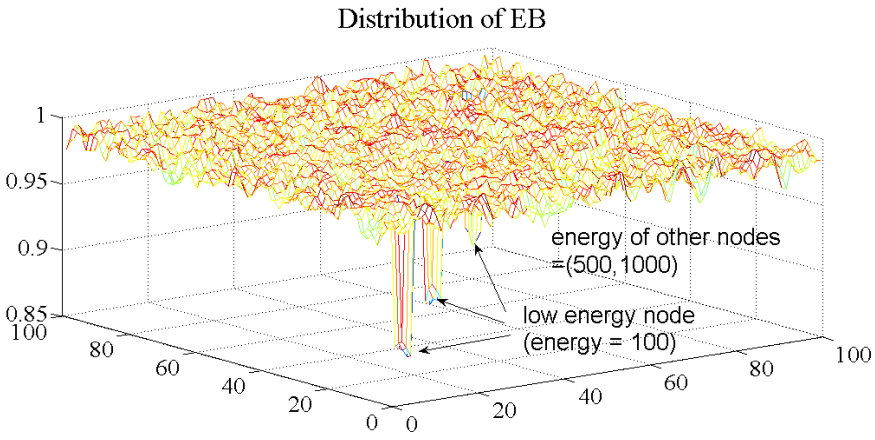


Figure 5.10 Distribution of Energy Balance (EB)

In Figure 5.10 the effectiveness of EB is shown. In a simulated environment in MATLAB with 100x100 sensor network grid, each node has maximum 8 neighbors. Only three nodes in the network have energy value of 100. But other nodes have energy between 500 to 1000. Then the distribution of locally computed EB across the network is shown in Figure 5.10. It can be observed that EB is high enough (very close to 1) everywhere, except in the neighboring region of the nodes with low energy. Therefore in a distributed network EB is a meaningful indicator of region with significant energy imbalance. Lower EB indicates higher degree of energy imbalance. The advantage of *ActRouting* in the initial example is described in Figure 5.8 with calculated EB metric.

Relay Selection: Now we describe the data forwarding scheme. The nodes in 1-hop neighborhood periodically share their normalized remaining energy (re_k) and Energy Balance metric EB_k ($k \in$ neighbor) through beacon message broadcast. From the $re_k(t)$'s of neighbors and its own remaining energy $re_i(t)$, each node i recomputes its metric $EB_i(t)$ and shares its current value of $EB_i(t)$ with neighbors using broadcasted beacon message. Suppose at time t, node i has data packet

$P_i(t)$ to send to the next relay node. Then energy balanced data delivery is performed as follows. (i) Having information of EB for all the neighbors, node i chooses the neighbor node with maximum EB, which has the same or less hop count to base station, as the next relay node to forward the data to. (ii) An opposite approach is used in relay selection when the remaining energy of the node is critically low. Then node i chooses the neighbor node as relay, which has minimum EB (in a hope to find a local region with some high remaining energy nodes) and same or less hop count to base station. (iii) In case the EB of the neighbors are very close to each other, the relay is selected as the one with maximum remaining energy, which has the same or less hop count to base station.

In addition *ActRouting* uses more activity-awareness in following way. A node outside currently active or predicted active region, has higher priority of getting selected as relay node (than a node inside region of active or predicted data source). Overall, in a completely distributed manner the proposed algorithm ensures relatively uniform distribution of energy consumption rate and hence better network lifetime. The strength of the proposed algorithm is that locally computed EB has inherent visibility into energy distribution in 2-hop neighborhood. To ensure faster routing convergence an added technique can be used. If a data packet is forwarded to nodes with same hop count for certain number of times, in the next step it is forwarded to the node with only lesser hopcount and better EB.

Algorithm 6 *ActRouting*: Activity-aware and Energy-balanced Routing

---

Each node i periodically computes the Energy Balance (EB) involving all of its 1-hop neighbors and itself. If node i has data packet to forward, select node j as its relay node as follows:

if $(EB_k^{min}/EB_k^{max}) \geq 0.9$ $(k \in neighbor_i)$ **then**
  $j = max(remaining\ energy)\ AND\ hop_j \leq hop_i$
else
  if $energy_i \leq energy_{critical}$ **then**
    $j = min(EB)\ AND\ hop_j \leq hop_i$
  else
    $j = max(EB)\ AND\ hop_j \leq hop_i$

Readjustment of radio duty cycle: For a data forwarding node, once the neighbor relay node is selected using *ActRouting*, it will readjust the duty cycle of the relay. This can keep the

selected route active so that the stream of data from the current source can be sent to base station with reliability and low latency. Then, once a node *i* selects its relay as node *j*, *i* repeatedly unicasts a message to *j* (until acknowledged) notifying it to run the radio on maximum duty cycle set by the application. To note that when current data source changes, *j* will not receive any data to forward for a number of radio sleep/wake intervals. Then *j* will run the radio back to the duty cycle decided by *ActDutyCycling*.

The role of *ActRouting* in the whole *ActiSen* system and the relationship of *ActRouting* with sensing and radio duty-cycling components is illustrated in Figure 5.3.

5.6 *ActiSen* System Design

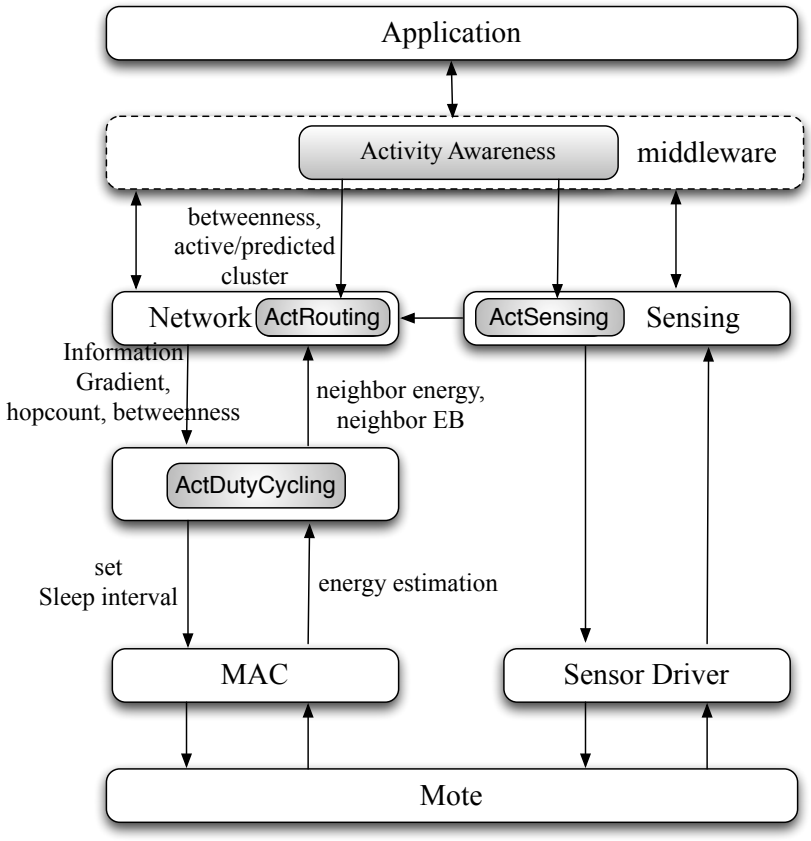
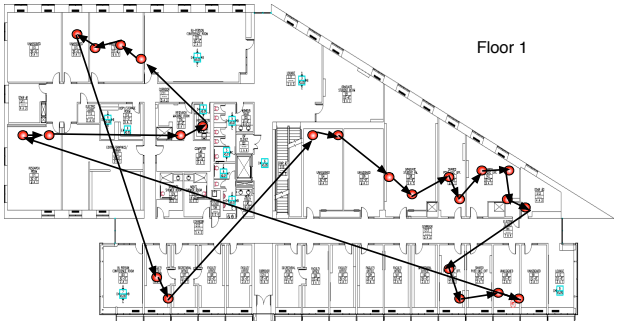


Figure 5.11 TinyOS software structure of *ActiSen*

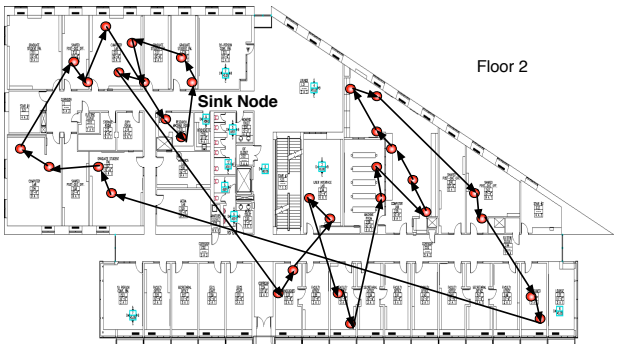
The *ActiSen* system is designed in TinyOS-2.x. The TinyOS software architecture in *ActiSen* is shown in Figure 5.11. The knowledge of Activity-Awareness resides in the middleware that can be utilized by application for smart environment. The cluster membership information is configured from Activity-Awareness module to the *ActRouting* module in network layer to dynamically configure the current gradient, predicted gradient and betweenness parameters. Based on these parameters the *ActDutyCycling* module configures link layer for dynamically configuring the sleep interval. The *ActSensing* module in sensing component uses activity knowledge from Activity-Awareness module and accordingly configures duty cycle for the activity detection sensors. The energy consumption is calculated using the relevant model of radio transmission, reception and radio idle states. The values used in the energy model are as follows: CC2420 radio parameters (19.7 mA current consumption in receive mode, 17.4 mA current consumption in transmit mode), 250 kbps data rate with 48 kByte data packet size, and the MSP430 MCU parameters (3 mA current consumption in active mode due to sensing and computation). The remaining node energy is accordingly updated. The node remaining energy information is used in the *ActRouting* network module in terms of normalized remaining energy and Energy Balance parameters. The information gradients, hopcount, remaining energy, Energy Balance (EB) are shared among neighbors, piggybacked in the broadcasted beacon message. Instead of beacon message of MAC protocols, the software design uses beacon message in routing protocol layer (in route and neighbor maintenance routine). The beacon message is broadcasted with (adaptive time period) among the neighbors. We have utilized the existing beacon message used in the routing layer. These beacon messages are piggybacked with more information containing information gradient, hop count and other required parameters. The beacon messaging in *ActiSen* uses dynamic and adaptive beacon period.

5.7 Performance Evaluation and Analysis

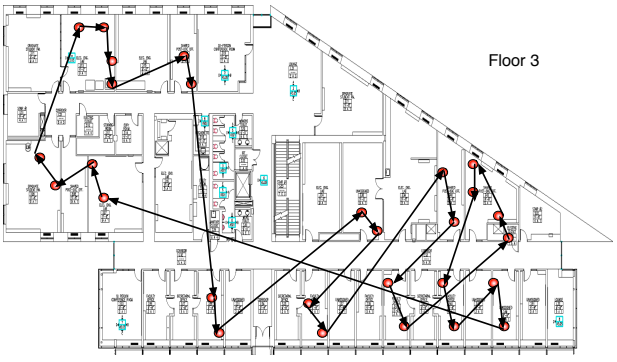
In this section we represent in detail the experimental evaluation and performance analysis to show the effectiveness of our proposed activity-aware design of *ActiSen*.



(a)



(b)



(c)

Figure 5.12 Most frequent activity sequences (order of active nodes) occurred in each floor of Motelab testbed during experiment

5.7.1 Experiment in Large Scale Real Sensor Network Tested

Evaluation environment: We have evaluated our proposed *ActiSen* system in large scale 82 node network of *TelosB* motes in Harvard *Motelab* sensor network testbed [58]. The experiments are conducted in 82 node network physically distributed in three floors, as shown in Figures 5.12(a), 5.12(b) and 5.12(c). The default radio range of used *TeloB* motes (uses CC2420 radio chip) is typically 50m for indoors and 125m for outdoors. The experiment environment in *Motelab* testbed is indoor environment.

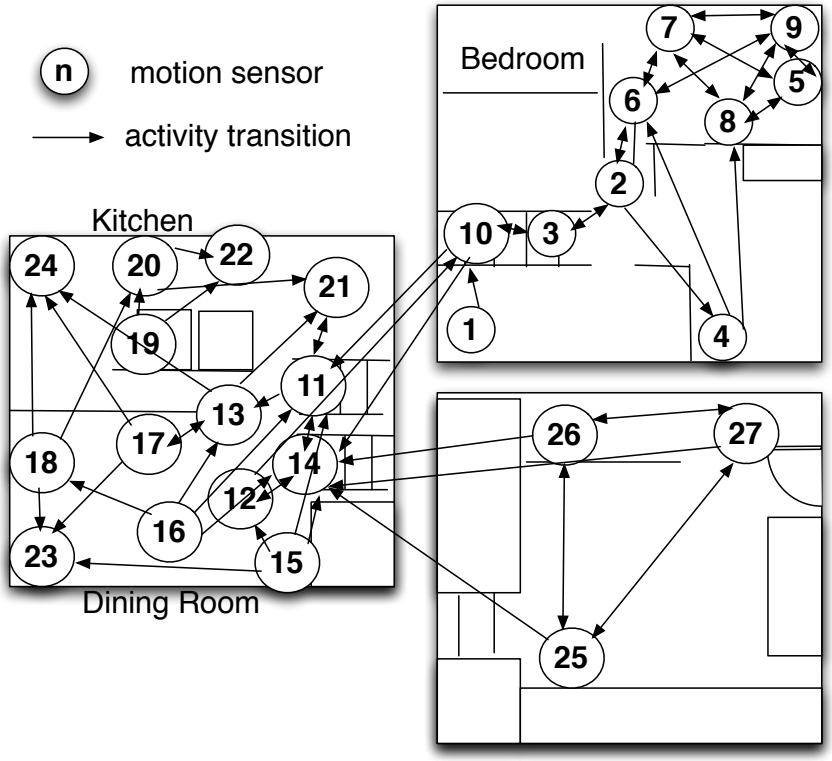


Figure 5.13 Activity Transition Probability Graph with 27 nodes, learnt from the CASAS Smart Home testbed [5]. Transition probability for example from node 27 to nodes 14, 25 and 26 are 12%, 45% and 40% respectively

Activity transition and data generation: From a separately deployed motion sensor network testbed we have learned the activity transition patterns and have validated the construction of ac-

tivity transition graph $ATPG$ (Figure 5.13). The activity transition patterns are modified to be scalable for a 82 node Motelab testbed, and is injected in the testbed for activity event generation and activity transition. The activity transition decides the order with which nodes will be active.

The activity event generation makes node(s) active, letting it send data to sink node (base station) at a very high rate (we used data sending rate of 480 Bytes/second). We have emulated the activity events by generating three independent sequences of active nodes (indicating motion trails) each in one of the three floors. From a remote server, periodic serial message (containing new active node numbers) is sent to the sensor motes in the testbed to generate the activity sequences. The sensor nodes receiving the serial message with it's ID start generating sensor data. Other nodes act as relay only. This periodic activation of nodes through serial message follow the activity transitions defined in the corresponding $ATPG$. In this way the activity transition experiments are performed with network-wide data collection. In addition each node periodically sends one local status data packet (containing information of remaining energy, duty cycle, hop count etc.) to sink every 30 seconds.

Comparison: For performance comparison we have compared combinations of existing routing schemes with selected MAC protocol, with *ActRouting* (*ActiSen* without *ActDutyCycling*) to show performance improvement due to *ActRouting*, and then compared with whole *ActiSen* system (*ActRouting* + *ActDutyCycling* on top of selected MAC protocol) to show performance improvement due to both activity aware routing and duty cycling. Following relevant routing protocols are used: $PMin$ (shortest path routing), CTP [59] (very commonly used data collection protocol for sensor networks, that uses link and path quality), and $CMAX$ (an energy aware protocol [34] where data is forwarded preferably to neighbor with higher remaining energy in the neighborhood). As explained earlier, *ActDutyCycling* in *ActiSen* system doesn't propose a new MAC protocol, but offers performance improvement of the MAC protocol in use by adapting the effective sleep interval with activity patterns. In the real testbed of $TelosB$ motes we have used the $X-MAC$ [22] as the base MAC protocol. In the experiments we have compared the following configurations:

(a) *PMin + X-MAC*: $PMin + X-MAC$ protocol.

(b) *CTP + X-MAC*: $CTP + X-MAC$ protocol.

(c) *CMAX + X-MAC*: $CMAX + X-MAC$ protocol.

(d) *ActRouting + X-MAC*: $ActRouting + X-MAC$ protocol.

(e) *ActiSen*: $ActRouting + ActDutyCycling$ (using selected $X-MAC$ protocol).

The 82 node network formed a 9 hop routing tree with -5 dBm transmission power of CC2420 radio of TelosB motes. The sink node is in middle of the three floors, as shown in Figure 4.8(b). In this network data collection scenario we have evaluated following parameters: (i) data delivery latency, (ii) data throughput, (iii) minimum node energy in the network through time (indicating network lifetime), (iv) duty cycle of the nodes, (v) network energy consumption etc. For radio duty cycling X-MAC is used with 5 seconds sleep interval, while *ActDutyCycling* (used on top of X-MAC), uses sleep interval range between TS^{max} = 10 seconds and TS^{min} = 1 second.

5.7.2 Performance Evaluation

Now we describe the performance analysis of our proposed *ActiSen* system compared to existing protocols.

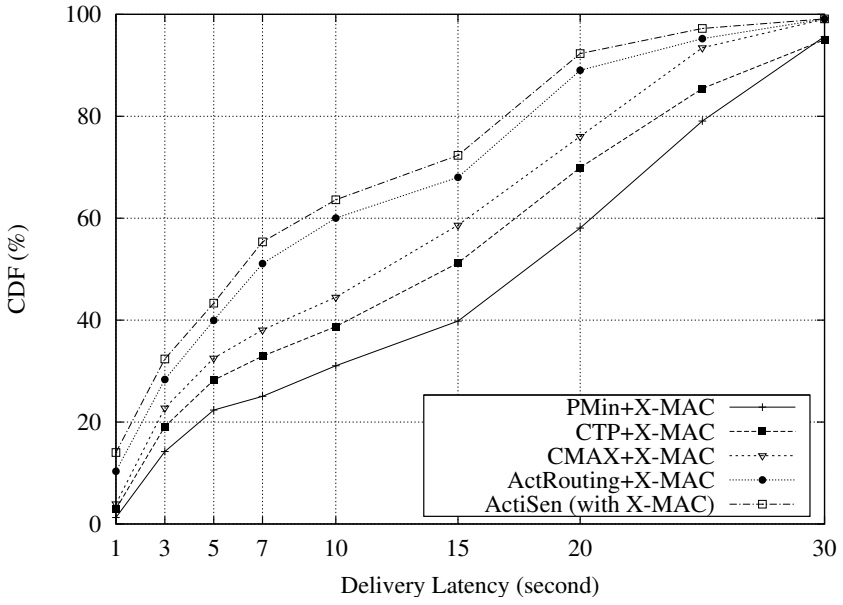


Figure 5.14 Distribution of data delivery latency

Latency: Figure 5.14 represents the distribution of delivery latency of packets in the 82 node network. It can be observed that *ActRouting + X-MAC* provides much lower latency than each of the comparing protocols (*PMin + X-MAC*, *CTP + X-MAC*, *CMAX + X-MAC*), and then the whole *ActiSen* system performs even better. In *PMin + X-MAC*, *CTP + X-MAC*, *CMAX + X-MAC*, 70% of the packets are delivered with latency between 20 seconds to 25 seconds. But in *ActRouting + X-MAC* and *ActiSen*, the 70% of the packets are delivered within latency around 13 seconds to 15 seconds. Therefore *ActiSen* provides much lower delivery latency, providing better performance to the application. *ActiSen* achieves this improvement in latency by (a) avoiding selection of currently active nodes (which are busy with sensing and sending own data) as relays by *ActRouting*, and (b) adaptive increase in duty cycle from *ActDutyCycling* for the active and predicted active nodes. Even in case the activity transitions don't always follow the *ATPG*, reactive duty cycle adjustments in *ActRouting* and *ActDutyCycling* provides improved latency.

Data Throughput: Figure 5.15 shows the data throughput for each node at sink. More throughput indicates more event data successfully delivered and reported at sink. It is observed that for each of the 82 nodes, *ActiSen* provides much improved data throughput than others. For all the 82 nodes *ActiSen* provides a data throughput improvement ranging from 5% to 13%. This advantage in *ActiSen* comes from (a) higher duty cycle on the route containing nodes between active data sources and the sink, and (b) avoiding selection of currently active nodes (which are busy with sensing and sending own data) as relays by *ActRouting*.

Lifetime: Figure 5.16 represents the minimum energy of any node in the entire network through time. This property decides the network lifetime. The energy consumption is configured in such a way that all nodes start with energy 2200 mAh. For indicating the rate of drop in minimum remaining node energy in network, Figure 5.16 represents the energy drop with respect to a chosen value of 0.05806 mAh. This value is chosen for purpose of analysis because the minimum node energy in $PMin + X - MAC$ reduces by an amount of nearly 0.05806 mAh energy in experiment run time of 1800 seconds (i.e. 30 minutes). This chosen value for analysis doesn't affect the nature of energy consumption of network. Now it can be observed that the minimum energy of any node in *PMin + X-MAC*, *CTP + X-MAC* and *CMAX + X-MAC* depletes faster than the one in *ActiSen*.

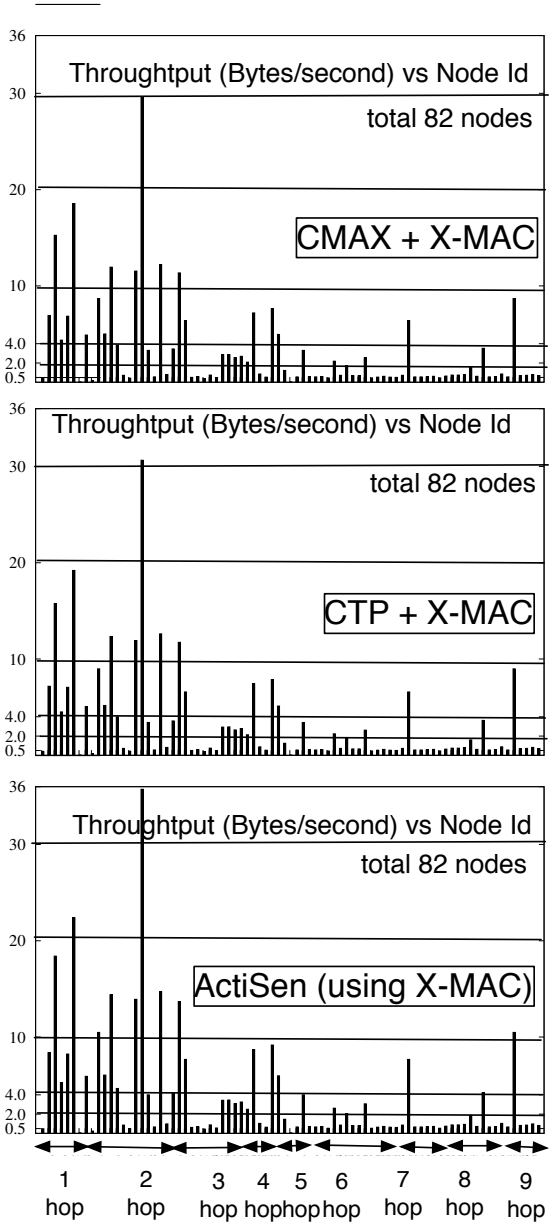


Figure 5.15 Data throughput at Sink

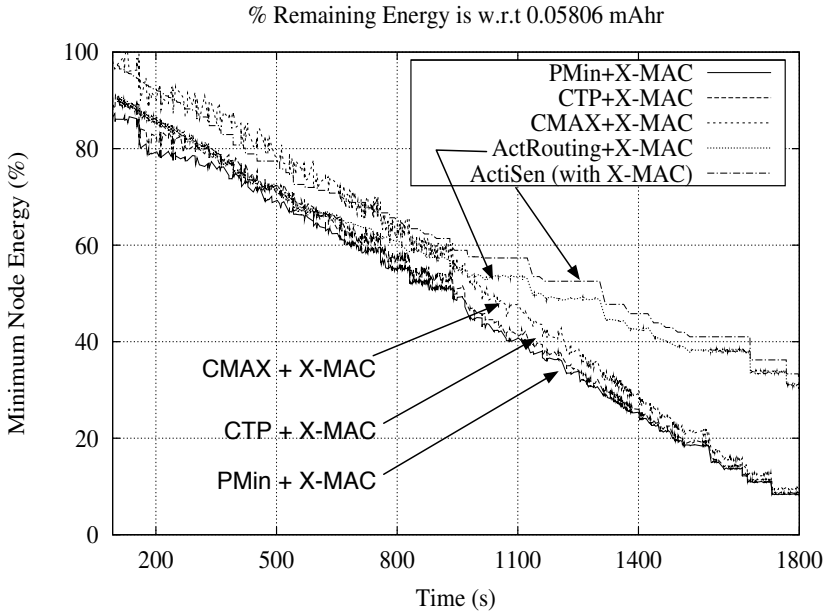


Figure 5.16 Minimum node energy in network

Within 1800 seconds time, the minimum energy of any node in *PMin + X-MAC*, *CTP + X-MAC* and *CMAX + X-MAC* reduces by an amount close to 0.05806 mAh. Therefore in protocols other than *ActiSen* network node depletes almost all it's energy within time 1800 seconds, had the network start with 0.05806 mAh for all nodes. But in same scenario in same time the minimum energy of any node in *ActiSen* would have still around 33% energy left. Therefore it is clear that network lifetime for *ActiSen* will also be much higher than others. *ActiSen* achieves this advantage because of (i) *ActDutyCycling*'s adaptively lowering of duty cycle during non-active phase of nodes and (ii) *ActRouting*'s energy balanced relay selection.

Dynamic duty cycling: Figure 5.17 shows an example of dynamic configuration of sleep interval (therefore dynamic change of duty cycle) of a node because of *ActDutyCycling* protocol in *Actisen*. The dynamic duty cycle in *Actisen* is illustrated with the example of activity transition from node 126 to node 129 to node 130 (126 $\rightarrow$ 129 $\rightarrow$ 130). During the experiment the node 126 was maintaining a lower effective duty cycle. But according to *ATPG* when predicted source is closer to that node its duty cycle gradually increases because of the information gradient. Then the node becomes the predicted source and increases the duty cycle to around 15%. It continues

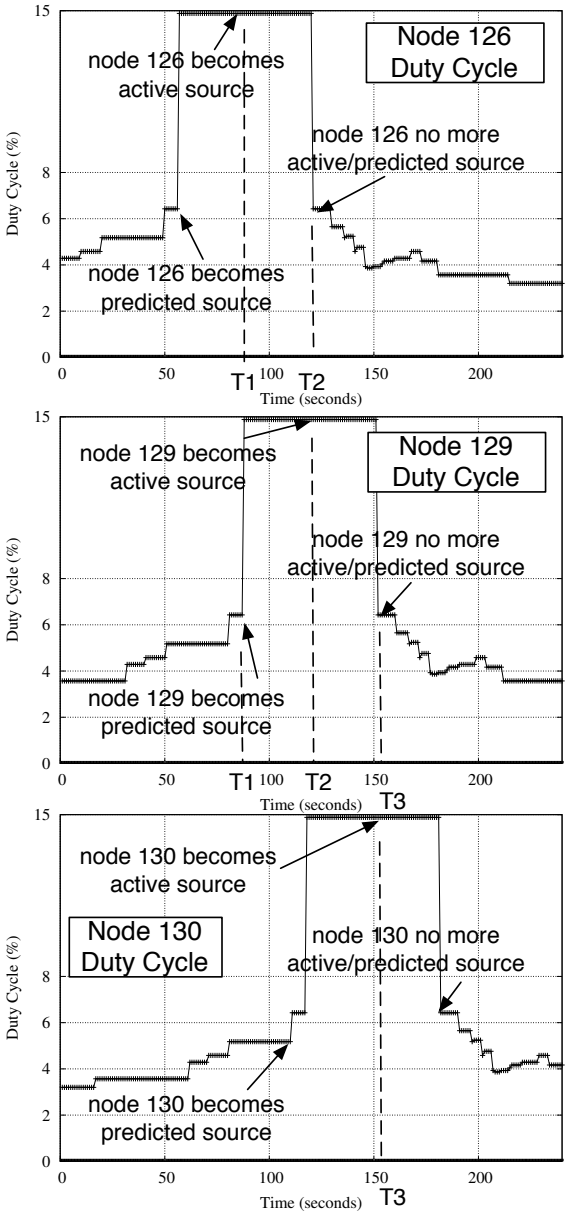


Figure 5.17 Dynamic radio duty cycle due to *ActDutyCycling*

the duty cycle of 15% since it turns into active source after some time. At the same time node 129 becomes predicted source and it's duty cycle gets higher. But when the node 126 is no more an active, the duty cycle gradually drops to low value. Then node 129 becomes active source and node 130 becomes predicted source. In similar manner as node 126, the duty cycle of nodes 129 and 130 gets updated. This shows an example of duty cycle change during activity transition in the network.

5.7.3 Experiment in Simulation

In TOSSIM sensor network simulator set up with different network size, the motion activity trajectory is simulated by periodically activating (setting nodes as the data source) a trajectory of nodes one by one with an interval of 30 seconds. In the simulation physical separation between the closest nodes was 5 meter. The radio range was set to the default, and each node had multiple neighbors, more than just the physically closest neighbors. For each network configuration, we have evaluated the following parameters: (i) mean delivery latency, (ii) data throughput at sink, (iii) projected network lifetime, and (iv) network energy consumption.

Evaluation environment: We have evaluated our proposed *ActiSen* system in TinyOS sensor network simulator TOSSIM [53] with varying network size from 20 to 100, to validate the scalability of proposed activity-aware design. For each network size, the nodes are deployed randomly in the area so that the minimum nodes separation is at least 5 meters. The sink node is at one corner of the deployed area. The simulation scenario is set up like a smart environment where detected activity information is reported to the sink.

Activity transition and data generation: In TOSSIM simulation experiments a python program periodcially decides the active nodes and the activity transitions. This controls the order of active nodes in the network.

Comparison protocols: For performance comparison, following relevant routing protocols are used: $PMin$ (shortest path routing), CTP [59] (link and path quality aware routing), and $CMAX$

(energy aware routing). As explained earlier, *ActDutyCycling* in *ActiSen* system doesn't propose a new MAC protocol, but offers performance improvement of the MAC protocol in use by adapting the effective sleep interval. For MAC protocol in TOSSIM simulation we have used CSMA.

In the simulation experiments we have compared the following configurations:

(a) $PMin + CSMA$: $PMin$ (shortest path routing) + $CSMA$

(b) $CTP + CSMA$: $CTP + CSMA$

(c) $CMAX + CSMA$: $CMAX$ (energy aware routing) + $CSMA$

(d) *ActRouting* + $CSMA$: *ActRouting* + $CSMA$

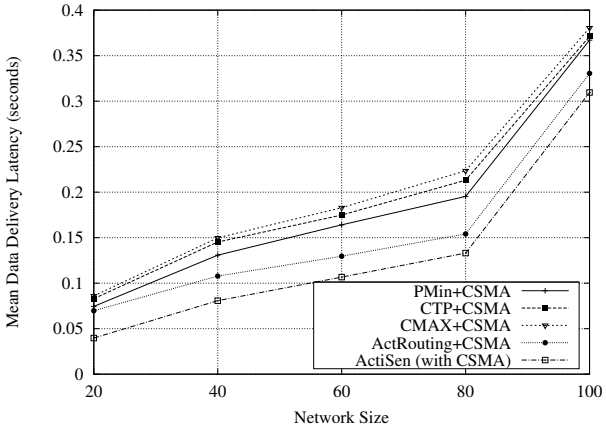
(e) *ActiSen*: *ActRouting* + *ActDutyCycling* with selected $CSMA$ protocol

5.7.4 Performance Evaluation

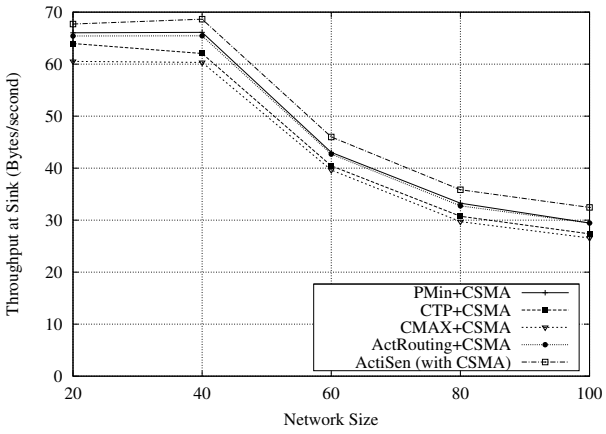
Data delivery latency: Figure 5.18(a) shows the mean data delivery latency for the selected data sources for varying network size. Now it is observed that *ActiSen* has much lower average latency than other for all network size. The improvement in latency ranges from 15% upto 46%. This shows that *ActiSen* provides better application performance with reduced latency in event data delivery and notification, which is also scalable from small to large network size.

Data throughput: Figure 5.18(b) shows the overall data throughput at sink for varying network size. It is observed that *ActiSen* has much higher data throughput than others for all network size. The improvement in throughput ranges from 2% upto 11%. This shows that *ActiSen* provides better application performance with improved data throughput in event data delivery and notification, which is also scalable from small to large network size.

Lifetime: Figure 5.19(a) shows that the network lifetime of network using *ActiSen* is significantly higher than that of others for all network size. *ActiSen* achieves improvement in network lifetime ranging from 17.1% upto 48.2%. The *ActRouting* and *ActDutyCycling* both contribute to this advantage. *ActDutyCycling* reduces the energy consumption of nodes when they are outside active region. *ActRouting* enables energy efficient selection of relay nodes on the active route. These lead to more balance in energy consumption of nodes in network. These all finally lead to effective decrease in node energy consumption in the network, resulting in improved lifetime.



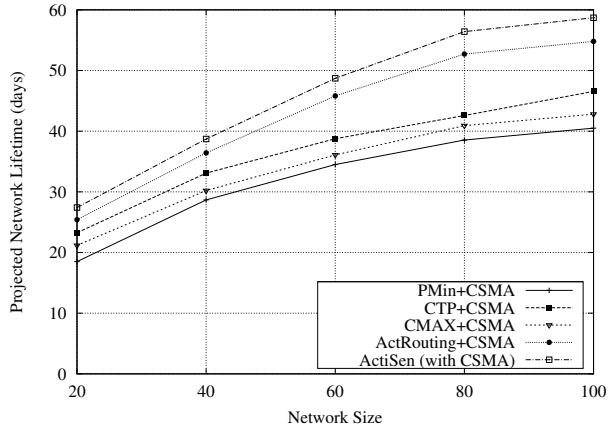
(a)



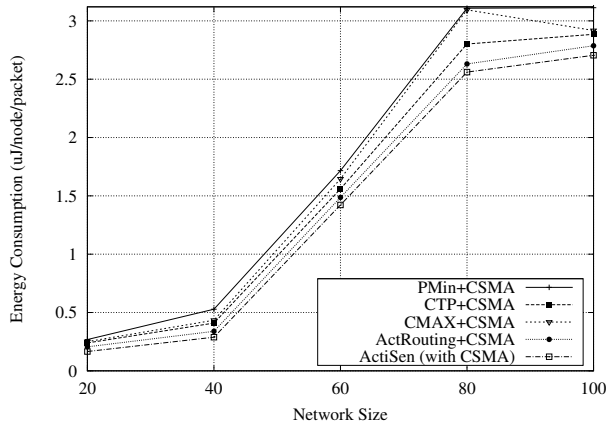
(b)

Figure 5.18 (a) Mean data delivery latency (seconds) with varying network size (b) Data throughput at Sink (Base Station) with varying network size

Network energy consumption: The improvement in duty cycle ensures reduced energy consumption for *ActiSen* for the network. This is shown in Figure 5.19(b) which shows average node energy consumption per successfully delivered packet at sink. This is a representative of network energy consumption which in turn denotes energy efficiency. It is observed that the average network energy consumption is much lower in *ActiSen* than others, and it is scalable with network size. This property of *ActiSen* gives the advantage of resource optimization for resource constrained sensor networks.



(a)



(b)

Figure 5.19 (a) Projected network lifetime (days) with varying network size (b) Energy consumption per successfully delivered message per node (uJ/packet/node) with varying network size

5.8 Summary

This chapter presents *ActiSen*, an activity-aware system design for wireless sensor networks, that can learn from detected activity patterns. The activity-aware design creates and updates an Activity Transition Probability Graph (ATPG) and then retroactively utilizes knowledge from it to dynamically configure the sensing, routing and radio duty cycling operations for providing: better performance to application, and resource optimization for resource constrained system. The activity-aware design achieves this using activity-aware sensing, activity-aware radio duty cycling, and activity-aware energy-balanced routing. The experimental results from real testbed and simulation experiments validate the advantages of activity-aware design and also show its scalability with varying network size.

PART 6

FINDINGHUMO: USER TRACKING IN SMART ENVIRONMENTS

In this chapter, we present *FindingHuMo* [4], a novel algorithm and system design for real-time and scalable tracking of multiple (unknown and variable number of) targets or human users in any crowded Smart Environments.

6.1 Background

Smart Environments are equipped with sensors which keep tracking the movements of users, who can for example be residents in a smart home, or employees in a smart workplace. Modeling the behaviors of users is a key step in developing particular applications in a Smart Environments. Identification and tracking the trajectories of users is the first step towards modeling. In many applications, e.g., in a smart workplace, a smart clinic, or a smart home, users may not want to reveal their identity all the time. In addition, the cost of sensors and communication device may drive designers to choose binary sensors that are relatively cost effective and more likely to be acceptable by general users. The binary sensors (e.g., a binary proximity based sensor, or a motion detector) only generates binary valued times series. This poses a challenge to identify and track user trajectories.

Our motivation in this work is solving two main challenges: (i) user specific motion tracking just from anonymous binary motion sensor data (binary motion sensors generate binary 0 or 1 samples, denoting no-motion and motion respectively), (ii) simultaneous tracking of multiple (unknown and variable number of) users in crowded environment where motion trajectories can overlap or crossover in all possible ways. Our designed system *FindingHuMo* (*Finding Hu*man *Mo*tion) does not rely on meticulous calibration, war-driving, GPS localization, or any form of fixed reference frame. The main contributions of our work are as follows:

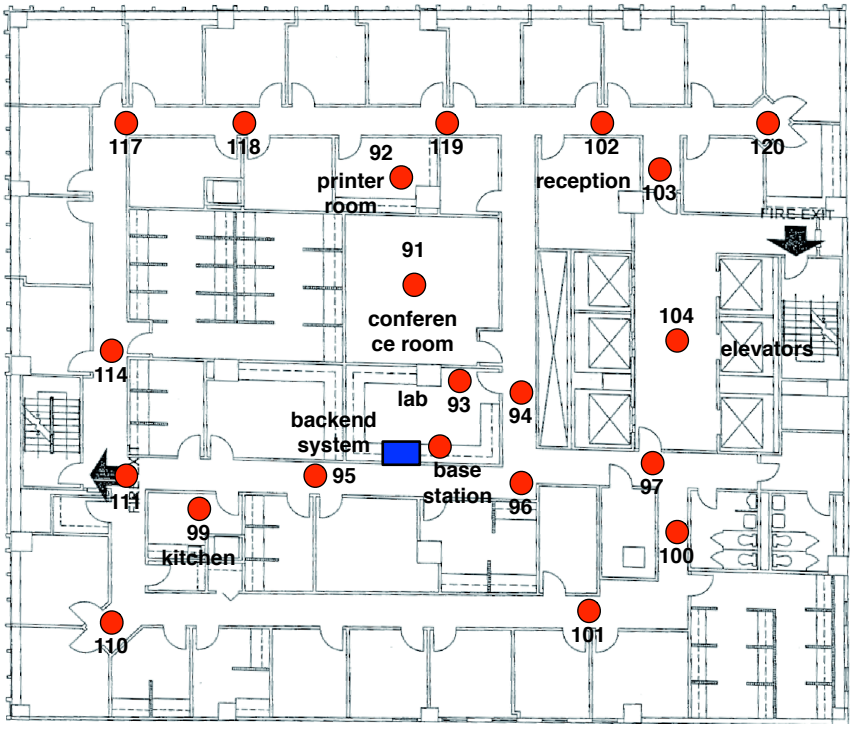


Figure 6.1 Motion sensor network deployment in a smart workplace environment. The sensor node position and node ID's are shown.

(a) design of a novel approach for scalable and real-time tracking of multiple targets from just anonymous binary motion data. This includes: (i) proposal of a motion activity context driven adaptive order Hidden Markov Model and Viterbi decoding (*Adaptive-HMM* algorithm), and (ii) an innovative path disambiguation algorithm (called *CPDA*). *Adaptive-HMM* removes the system noise and ambiguity in smaller time window using effect of hidden states, while *CPDA* removes path ambiguity in a larger time window by applying constraints and inference on user interaction based graph.

(b) Complete system design and performance evaluation in a real-time smart environment (environment is shown in Figure 6.1).

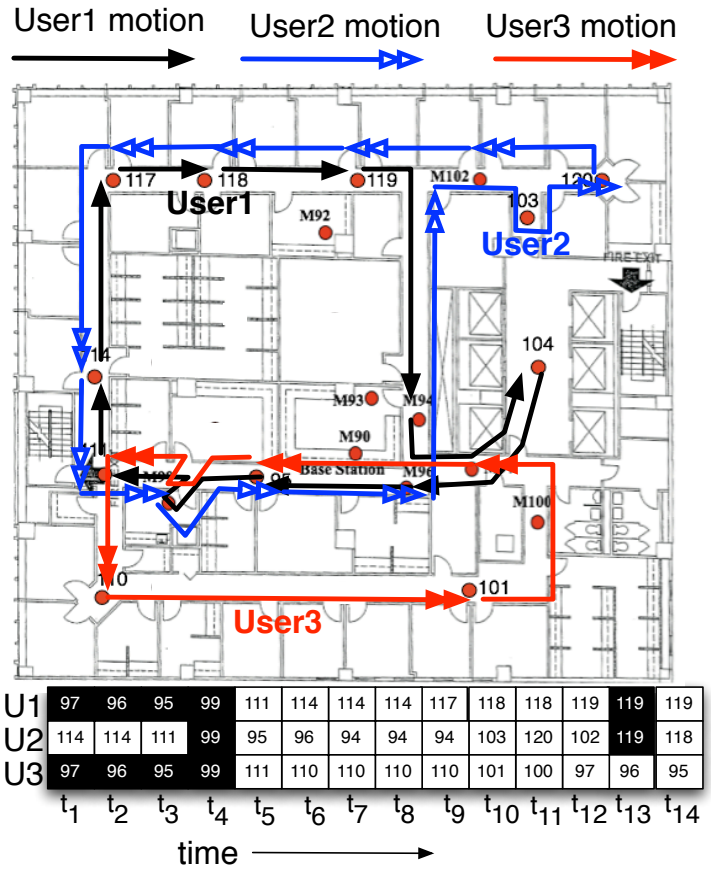


Figure 6.2 Multi-user overlapping motion trajectories. The table below the figure shows the node or state sequence of each of the 3 users User1, User2 and User3, with time. The dark blocks in the table indicate motion overlap or crossover among the users.

6.2 Proposed Real-Time Multi-Target Tracking System

The working methods of the whole system is described step-by-step with the physical layout as shown in Figure 6.1 and accompanying example scenario in Figure 6.2. Figure 6.2 explains instance of overlapping multi-user motion trajectories in a real smart workplace environment (layout in Figure 6.1).

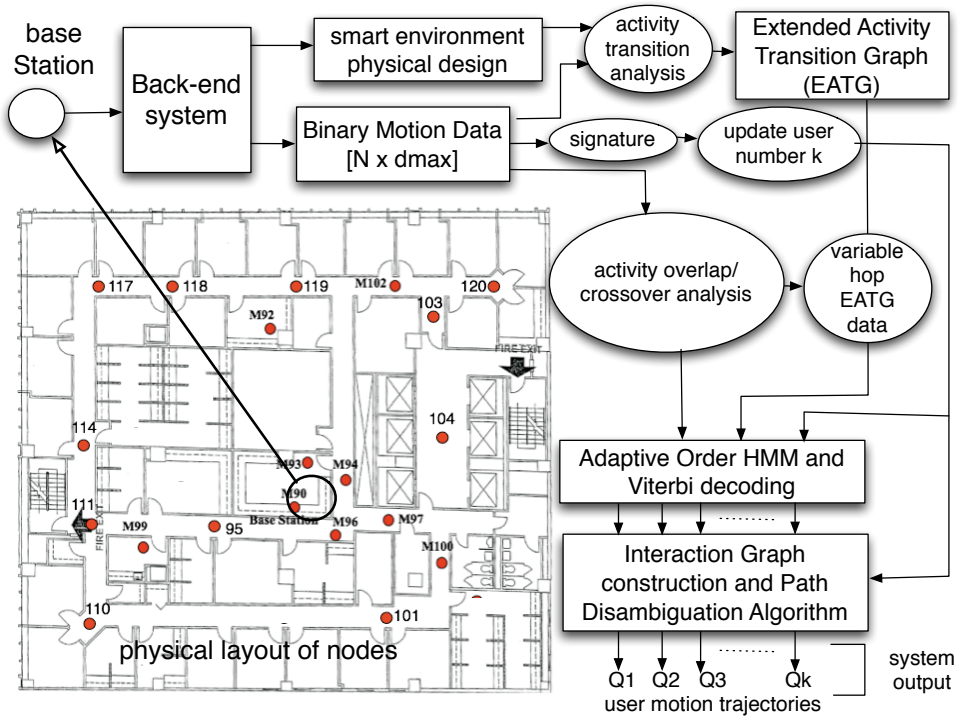


Figure 6.3 *FindingHuMo* system: Multi-target tracking from binary motion sensor network.

Methodology **System Resources:** *FindingHuMo* system consists of: (i) a static wireless sensor network (with binary motion sensors) deployed throughout the physical environment. The binary motion data from each sensor node are collected through multi-hop network into a base station; (ii) a back-end system computing user tracking algorithms on collected binary motion datastream.

Table 6.1 *List of parameters and their description*

M	Set of motion sensor nodes ($N = \|M\|$)
$b_j(t)$	Output of sensor node m_j at timeslot t
K	The number of moving users
$q_i(t)$	Motion status of user i at timeslot t
$(t_s, (t_s + d_{max}T))$	Time window of *Adaptive-HMM* compuation
$(t_s, (t_s + Cd_{max}T))$	Time window of *CPDA* compuattion

System Model and Problem Definition: Suppose the set of motion sensor nodes is M (N=$|M|$). Then the motion sensor network collects binary motion data $B(t) = \bigcup_{j=1}^{N} b_j(t)$, where $b_j(t) = 0$ or 1, is the binary motion status detected by node m_j ($\forall\ j \in (1, N)$) at timeslot t. Now each m_j ($\forall\ j \in (1, N)$) denotes the possible motion states of a user in the environment. Therefore for user i ($i \in (1, K)$, K is the number of users), if the motion activity state at time t is $q_i(t)$=m_j, it indicates that the user i is near the location of node m_j at time t. The application output of the system is thus the sequence of states $\{q_i(t_0), q_i(t_0 + T), q_i(t_0 + 2T), ..., q_i(t_0 + d.T), ..., q_i(t_0 + (D - 1)T)\}$ $\forall\ i \in (1, K)$, where D can be a time period of user tracking (say 24 hours). The real-time requirement is that each $q_i(t_0 + dT)$ has to be computed in time $(t_0 + dT, t_0 + dT + D'T)$ where $D' << D$. Then the research problem is how to compute motion trajectories (the sequence of states) dynamically for all the users, from anonymous binary motion data stream B which doesn't contain any user specific information but just the collective motion status in the environment.

System Procedure: The operational architecture of proposed *FindingHuMo* is shown in Figure 6.3. Based on some motion signature activities the system increments / decrements K, the current number of users. Then based on detection of motion non-overlap /overlap in the binary motion data of time window t_S to $(t_S + d_{max}T)$, the system applies a variable state and variable order modified HMM. This exploits more information available and thus can capture contexts of user activities more accurately. The output is segments of state sequences s_i ($1 \leq i \leq K$) for K users. This, combined (or can be called *stitched*) with decoded path segments in a larger time window of length $d_w.T$ (where $d_w.T$=$C.d_{max}T$, C is a constant), generates an *Interaction Graph*. A proposed path disambiguation algorithm *CPDA* is processed on that graph, which finally results in disambiguated node or state sequences of individual users. Algorithm 7 provides pseudo code

for the back-end system. The description and pseudocode of *Adaptive-HMM* and *CPDA* are later presented in following subsections.

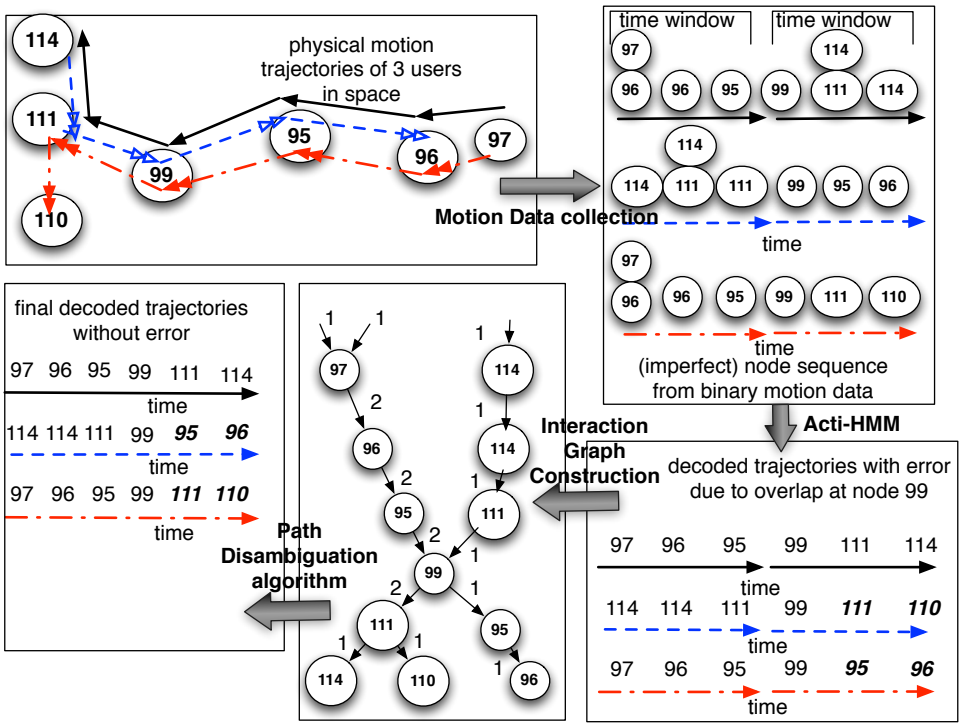


Figure 6.4 A working example of *FindingHuMo*.

Working Example: Figure 6.4 illustrates how proposed *FindingHuMo* solves the above-mentioned problem. The collected raw motion data contains unreliable node sequence with system noise. This is refined by applying *Adaptive-HMM*. The decoded state sequence may still contain error due to path crossover (e.g. crossover of decoded path for user 2 and user 3 at node 99). This is further corrected by *stitching* the decoded paths and forming an *Interaction Graph*, which is then disambiguated by applying proposed *CPDA* algorithm. This results in final decoded motion trajectories. It is worth mentioning that the position of user is presented in form of sensor nodes' position. Thus the tracking accuracy will be more (w.r.t the actual physical location of user) if the sensor node deployment is more dense. Also the maximum number of users that can be tracked simultaneously, is bounded by the number of sensor nodes deployed.

Next we separately present our proposed algorithms *Adaptive-HMM* and *CPDA*.

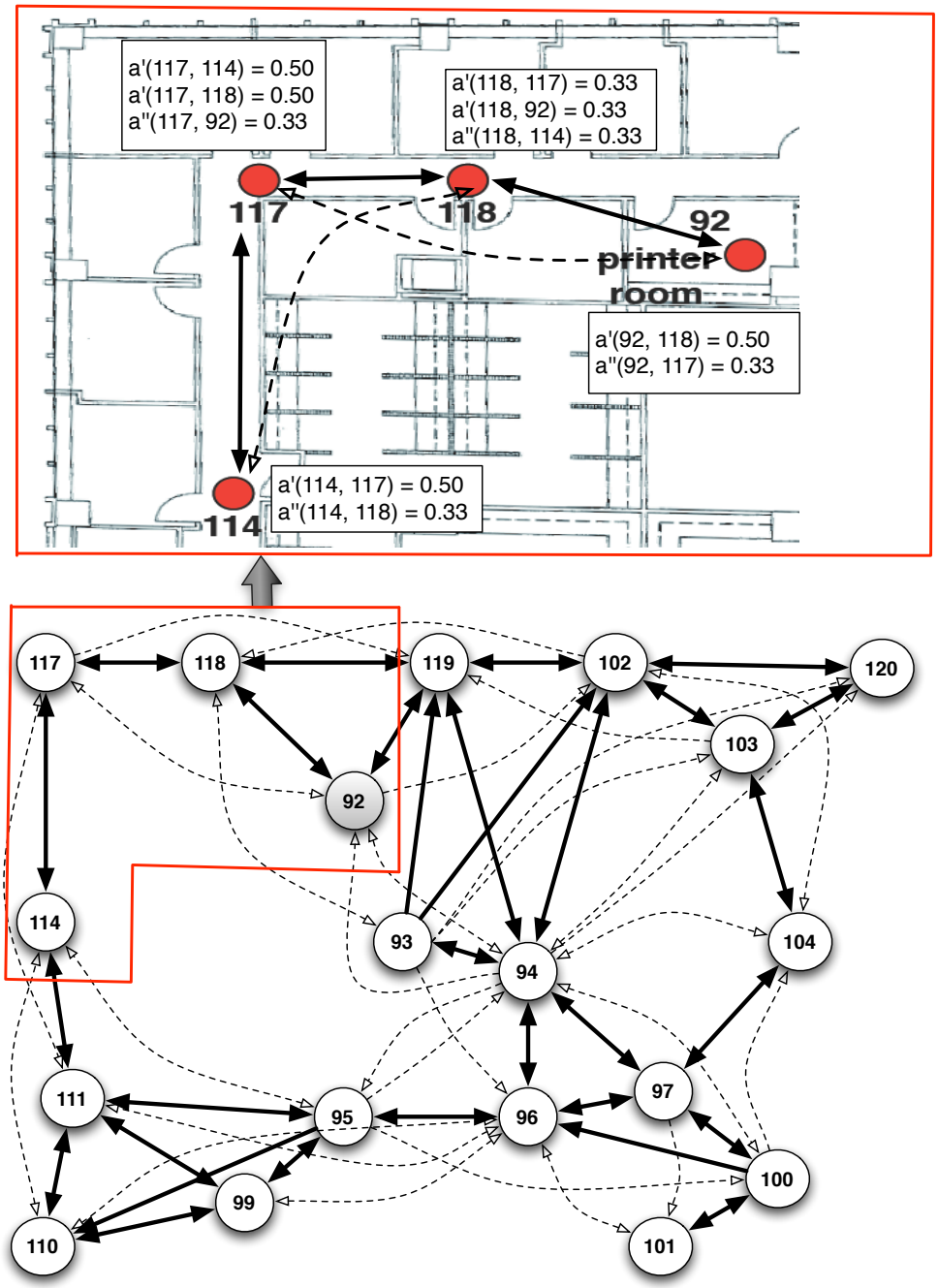


Figure 6.5 Extended activity transition graph *EATG* constructed for the smart environment layout shown in Figure 6.1. Solid lines indicate activity transition between nodes 1-hop away, while dashed lines indicate activity transition between nodes 2-hop away. Nodes 1-hop to each other, can be physically reachable without triggering any other node.

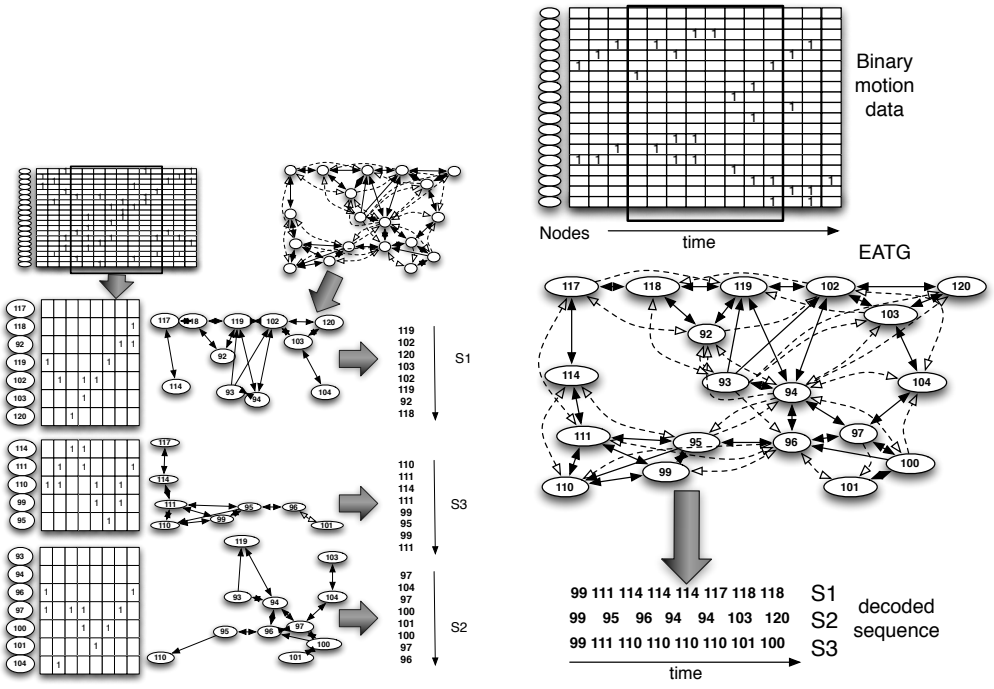


Figure 6.6 (a) *Adaptive-HMM*: Splitting of non-overlapping motion into individual HMM's and then decoding state sequences using first order HMM. (b) *Adaptive-HMM*: Decoding of state sequences for overlapping motion in larger state and second order HMM.

Algorithm 7 Pseudo code for back-end system connected to base station of sensor network

Input: Binary motion data $B(t_s, t_s + C.d_{max}.T) = \bigcup_j b_j(t)$ where $t_s \leq t \leq (t_s + C.d_{max}.T)$ and $j \in M' \subset M$

Output: Decoded state sequence $Q(t_s, t_s + C.d_{max}.T) = \bigcup_i \{q_i(t_s), ..., q_i(t_s + C.d_{max}.T)\}\ \forall\ i \in (1, K)$

1: Q_w = NULL;
2: **for** $t = t_s \rightarrow t_s + C.d_{max}.T$ **do**
3: $Q'(t, t + d_{max}.T) = Adaptive\text{-}Hmm(B(t, t + d_{max}.T))$; (Adaptive-HMM algorithm)
4: $Q_w = Q_w \bigcup Q'(t, t + d_{max}.T)$;
5: $t \leftarrow t + d_{max}.T$;
6: $Q(t_s, t_s + C.d_{max}.T) = CPDA(Q_w)$; ($CPDA$ path disambiguation algorithm)

Algorithm 8 Pseudo code for *Adaptive-HMM* algorithm *Adaptive-Hmm*()

Input: Binary motion data $B(t, t + d_{max}.T)$

Output: Decoded state sequence $Q(t, t + d_{max}.T)$

1: $EATG(B(t, t + d_{max}.T))$; (explained in subsection 6.2.1)
2: Update extended activity transition graph G;
3: $\lambda = FormHMM(A, C, d_{max}, \tau, \Pi)$; (Adaptive-HMM model creation, explained in subsection 6.2.1)
4: $K = UserCount(B(t, t + d_{max}.T))$; (to update the number of current users K, explained in subsection 6.2.1)
5: $Q(t, t + d_{max}.T) = Viterbi(\lambda, K)$; (explained in subsection 6.2.1)

6.2.1 *Adaptive-HMM* Algorithm

This subsection describes the *Adaptive-HMM* algorithm (*Adaptive-Hmm*() in main Algorithm 7). The pseudocode for *Adaptive-Hmm*() is shown in Algorithm 8. *Adaptive-HMM*'s operation is motion activity driven to some extent. This is in a sense that, based on the activity amount detected in the motion data segment, it applies different methods to extract the motion trajectories.

Extended Activity Transition Graph: This explains the task $EATG()$ in Algorithm 8. It is important to note that even single user trajectory, or multi-user non-overlapping trajectories cannot be reliably concluded from just the binary motion data. Some knowledge of activity transition relationship among the nodes (or states) is necessary for extracting exact motion trajectories. For example in Figure 6.1 if both the motion sensors 93 and 94 are triggered, then from transitional relation from last activated state 96 it can be concluded that the motion trajectory was ...96→94...

instead of ...96→93... The notion of activity transitional relationship among the nodes or states is presented and used in this work in the form of an Extended Activity Transition Graph or *EATG*. In *EATG* $G = (M, E', E'', A', A'')$, a node $m_j \in M$ represents a sensor node in the environment, weighted edge $e' \in E'$ denotes a pair of sensor nodes that can physically be reached directly from each other, and weighted edge $e'' \in E''$ denotes a pair of sensor nodes that can physically be reached from each other by triggering one more node in between. The weights a' and a'' of edge e' and e'' respectively denote direct 1-hop and indirect 2-hop activity transition between the nodes.

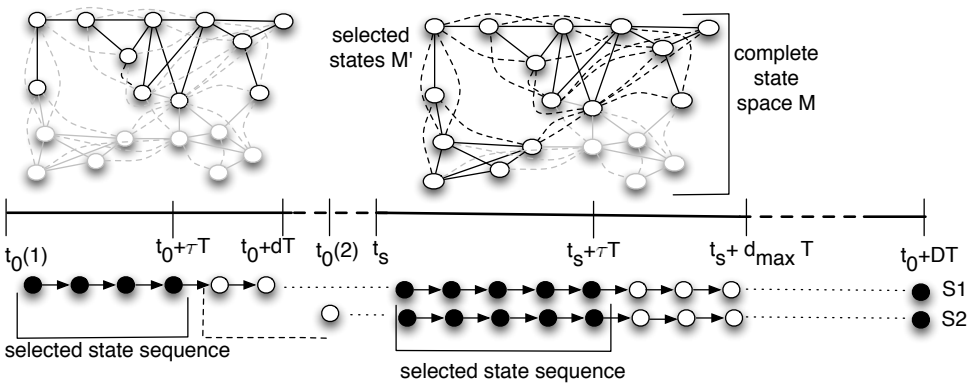
$$a'(m_{j_2}, m_{j_1}) = \frac{\#events(m_{j_2}(t) \Rightarrow m_{j_1}(t+T))}{\#events\, m_{j_2}(t)} \tag{6.1}$$

$$a''(m_{j_2}, m_{j_1}) = \frac{\#events\ (m_{j_2}(t) \Rightarrow m_{j_1}(t+2T))}{\#events\ m_{j_2}(t)} \tag{6.2}$$

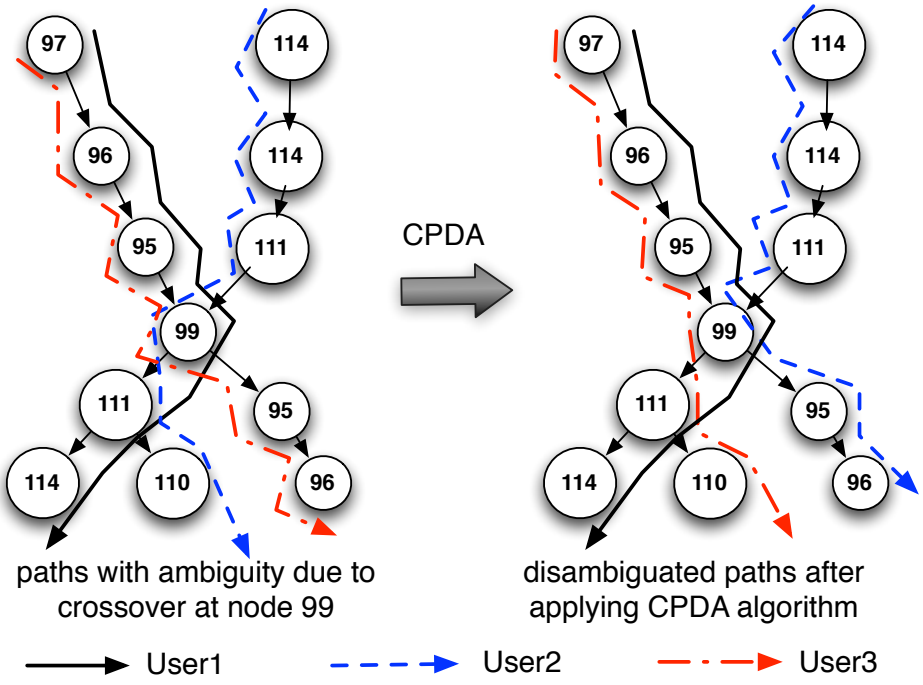
Example *EATG* belonging the layout in Figure 6.1 is shown in Figure 6.5. Direct transition probability A' is constructed either directly from the physical layout or is trained from collected binary motion data as illustrated in equation 6.2. Indirect 2-hop transition probability A'' is trained from binary motion data as in equation 6.2. The term $\#events\ m_{j_2}(t) \Rightarrow m_{j_1}(t+T)$ denotes the number of events where m_{j_2} is triggered at any time t and then followed by m_{j_1} at time $(t+T)$.

Adaptive Order HMM Modeling: This explains the task $FormHMM()$ in Algorithm 8. The system model is designed as a modified Hidden Markov Model (HMM) with a discrete time stochastic process. The modified model is named *Adaptive Order HMM* or *Adaptive-HMM*, and it's working is shown in Figure 6.7(a). The *Adaptive-HMM* model is $\lambda = (A, C, d_{max}, \tau, \Pi)$ and the set of states is M'.

(a) **States**: M'=$\{m_j\}$ where $m_j \in M$. So M' contains only part of the states in M. *Adaptive-HMM* chooses only the subset of states that are active and the neighbor (1-hop or 2-hop in *EATG*) states. This reduces the computational complexity without compromising the accuracy (theorem 3). In the HMM time window t to $(t + d_{max}.T)$, if the system detects say x non-overlapping motion (activated nodes at each slot of T are 1-hop away) it creates or forks out x HMM computations with each state set M' containing activated nodes of corresponding motion sequence and



(a)



(b)

Figure 6.7 (a) Activity context driven selection of state set and state transitions in *Adaptive-HMM*. The state sequence is saved only till $t_s + \tau.T$, and the next HMM window computation starts at $t_s + (\tau + 1).T$ instead of $t_s + (d_{max} + 1).T$. For single activated state the state set is smaller (activated nodes and their 1-hop neighbors) and uses transition only from $(t - 1)$. But for multiple simultaneous activated states the state set is larger (activated nodes and upto their 2-hop neighbors) and uses transitions from time $(t - 2)$ and $(t - 1)$. (b)Illustrative example of proposed *CPDA* algorithm.

their 1-hop nodes (example in Figure 6.6(a)). But if it detects overlapping motion (sequence of motion activated nodes overlap in at least one slot of T) it creates a single HMM computation with state set M' containing activated nodes of motion sequences and upto their 2-hop nodes (example in Figure 6.6(b)). *Example*: In the example in Figure 6.6(a), the state sequence 110→111→114→111→99→95 →99→111 generate non overlapping state sequences with other activated states due to other users. Thus the state set of individual HMM computation contains only activated nodes (110, 111, 114, 99, 95) and their 1-hop nodes (117,96,101). But when that user's motion states are overlapped with others (motion shown in Figure 6.2) then the HMM computation contains all the active states and upto their 2-hop neighbors (Figure 6.6(b)).

Sub-state selection in *Adaptive-HMM*: The sub-state selection in *Adaptive-HMM* doesn't affect the optimality of HMM model and Viterbi computation in our application scenario (due to activity transitional relationship among nodes upto 2 hop away).

Theorem 3 *In Adaptive-HMM the reduced state set M' results in the same optimal state sequence as that with complete state set M.*

(b) **State Transition Probability**: In HMM computation for non-overlapping motion, $A = \{a(j_2, j_1)\}$, where $a(j_2, j_1) = P[q(t) = m_{j_1} | q(t-1) = m_{j_2}] = a'(m_{j_2}, m_{j_1})$, and in HMM computation for overlapping motion $A = \{a(j_3, j_2, j_1)\}$, where $a(j_3, j_2, j_1) = P[q(t) = m_{j_1} | q(t-1) = m_{j_2} AND q(t-2) = m_{j_3}] = a'(m_{j_2}, m_{j_1}).a''(m_{j_3}, m_{j_1})$; $a(j_3, j_2, j_1)$ is the state transition probability from motion activated node m_{j_3} at $(t-2)$ and m_{j_2} at $(t-1)$ to node m_{j_1} at time t ($q(t)$ denotes the current motion activated node at time t). Equivalently $a(j_2, j_1)$ denotes state transition from $(t-1)$ to t. Here *Adaptive-HMM* is motion activity driven. If there is motion overlap within time window, A includes transition from states at $(t-2)$ and $(t-1)$. But for no overlap A includes transition only from state at $(t-1)$.

(c) **Emission Probability Distribution**: For non-overlapping motion $C = \{c_j(p)\}$ and for overlapping motion $C = \{c_{j_2 j_1}(p)\}$. $c_{j_2 j_1}(p) = P[o_p | q(t) = m_{j_1} AND q(t-1) = m_{j_2}]$ is the probability that the system outcome at time t is $o_p \subseteq M'$ given node m_{j_1} is activated at current t and node m_{j_2} was activated at $(t-1)$. Equivalent meaning stands for $c_j(p)$.

(d) **Time Window and Threshold**: d_{max} is the time duration of the applied HMM. So the HMM time length or time window is from t_s to $(t_s + d_{max}.T)$. The HMM is computed for the time window t_s to $(t_s + d_{max}.T)$, but the resulting state sequence are saved only from t_s upto an instant $(t_s + \tau.T)$. τ indicates a threshold point for accepting the resulting state sequence.

(f) **Initial State Distribution**: $\Pi = \{\pi_j\}$ $(m_j \in M')$, where $\pi_j = P[q(t_s) = m_j]$ is the probability that at starting time t of HMM time window the activated node is m_j.

User Count in HMM time window: This explains the task $UserCount()$ in Algorithm 8. It updates the number of users K in HMM time window $(t, t + d_{max}.T)$ using following: $K = \max(K_{pre} + Sig_{in} - Sig_{out}, K_{now})$. K_{pre} is the number K from the previous HMM window $(t - d_{max}.T, t)$, Sig_{in} is the number of *user entry signatures* (e.g. node sequence $104 \rightarrow 103$), Sig_{out} is the number of *user exit signatures* (e.g. node sequence $97 \rightarrow 104$), K_{now} is the maximum number of triggered nodes (that are at least 2 hops away in $EATG$) in any unit slot of T in the window.

Viterbi Computation: The procedure is explained in the task $Viterbi()$ in Algorithm 9. Given the values of M', A, C, d_{max}, τ and Π, the HMM generates system observation sequence $O = O(t)$ $O(t + T)$ $O(t + 2T)$ $...O(t + dT)..$ $O(t + d_{max}.T)$ = say $o^f(t, T, d_{max})$ (where each $O(t + dT) \subseteq M'$). So the problem is given such system observation sequence O, the model λ and states M', how to choose the corresponding state sequence $Q_i = q_i(t)$ $q_i(t + T)..q_i(t + dT)..q_i(t + \tau.T)$ = (say) $q_i^f(t, T, \tau)$ for each user i between time t and $(t + \tau.T)$. Finding the optimal state sequence with respect to the Maximum a posteriori (MAP) criterion is efficiently done with the Viterbi algorithm and tracing back through a matrix of back-pointers, starting from the end of the sequence. Standard Viterbi decoding algorithm is modified for: multiple observation, multiple sequence decoding, and fitting for activity awareness. For non-overlapping motion, viterbi algorithm is computed on first order HMM [38] (task $Viterbi^1()$) where transitions from time $(t - 1)$ to t are considered. For overlapping motion, viterbi algorithm is computed on second order HMM [63] (task $Viterbi^2()$) where transitions from time $(t - 2)$ and $(t - 1)$ to t are considered. The pseudocodes for $Viterbi^1()$ and $Viterbi^2()$ are shown in Algorithm 10 and Algorithm 11 respectively. It's worth mentioning that second order HMM captures a more amount of the activity contextual information than the first order HMM.

Algorithm 9 *Viterbi*(λ, K): Viterbi decoding in *Adaptive-HMM*

Input: HMM model λ, user number K

Output: Decoded sequence $Q(t, t + d_{max}.T)$

1: **if** No motion overlap detected among trajectories **then**
2: for each trajectory: (i) Update λ by keeping only triggered nodes and their neighbors in $EATG$; (ii) $Q(t, t + d_{max}.T) = Viterbi^1(\lambda)$; (1-state Viterbi decoding)
3: **else**
4: (i) Update λ by keeping only triggered nodes and their neighbors in $EATG$; (ii) $Q(t, t + d_{max}.T) = Viterbi^2(\lambda, K)$; (2-state Viterbi decoding)

Algorithm 10 Viterbi algorithm $Viterbi^1(\lambda)$ on HMM for $non-overlapping$ motion

Input:
HMM $\lambda=(A, C, d_{max}, \tau, \Pi)$;
state set M';
observation sequence $O=O(t_s)\ O(t_s + T)..O(t_s + dT)..O(t_s + d_{max}.T)$;
A and C contain state information from $(t-1)$ to t;
M' contains activated nodes and their 1-hop nodes in $EATG$.

Output: Optimal activity state sequence $Q=q(t_s)\ q(t_s + T)..q_i(t_s + dT)..q_i(t_s + \tau.T)$ $(\tau \leq d_{max})$.

Variables:

$$\delta_d(j)= \max_{q(t_s),q(t_s+T),..,q(t_s+(d-1)T)} P[q(t_s), q(t_s + T), .., q(t_s + dT) = j, O(t_s), .., O(t_s + dT)|\lambda]$$

$$\psi_d(j)= \operatorname*{argmax}_{q(t_s),q(t_s+T),..,q(t_s+(d-1)T)} P[q(t_s), q(t_s + T), .., q(t_s + dT) = j, O(t_s), .., O(t_s + dT)|\lambda];$$

1: Initialization:

$$\delta_0(j) = \pi_j.c_j(O(t_s))\ (m_j \in M') \quad and \quad \psi_0(j) = 0$$

2: Recursive step:

$$\delta_d(j) = \max_{m_{j'} \in M'} [\delta_{d-1}(j')a(j', j)].c_j(O(t_s + dT))$$

$$\psi_d(j) = \operatorname*{argmax}_{m_{j'} \in M'} [\delta_{d-1}(j')a(j', j)]$$

where $(1 \leq d \leq d_{max}$ and $m_j \in M')$

3: Termination:

$$P^* = \max_{m_j \in M'} [\delta_{d_{max}}(j)] \quad u^*_{d_{max}} = \operatorname*{argmax}_{m_j \in M'} [\delta_{d_{max}}(j)]$$

4: State sequence backtracking:

$$u^*_d = \psi_{d+1}(u^*_{d+1}), \quad d = (d_{max} - 1), (d_{max} - 2), ..., 0$$

5: Output: $\{q(t_s)=u^*_0,\ q(t_s + T)=u^*_1,\ ..,\ q_i(t_s + dT)=u^*_d..q_i(t_s + \tau.T)=u^*_\tau\}$

Algorithm 11 Viterbi algorithm $Viterbi^2(\lambda, K)$ on HMM for *overlapping* motion

Input:
HMM λ=(A, C, d_{max}, τ, Π);
state set M'; observation sequence O=$O(t_s)$ $O(t_s + T)..O(t_s + dT)..O(t_s + d_{max}.T)$;
A and C contain state information from $(t-2)$ and $(t-1)$ to t;
M' contains activated nodes, their 1-hop and 2-hop nodes in $EATG$.

Output:
Optimal activity state sequence Q=$q(t_s)$ $q(t_s + T)..q_i(t_s + dT)..q_i(t_s + \tau.T)$ for each $k \in K$ ($\tau \leq d_{max}$).

Variables:
$\delta_d(j', j, k)$= $\max\limits_{q(t_s),q(t_s+T),..,q(t_s+(d-2)T)} P[q(t_s), q(t_s+T), .., q(t_s+(d-1)T) = j', q(t_s+dT) = j, O(t_s), .., O(t_s+dT)|\lambda]$
for each user $k \in K$
$\psi_d(j', j, k)$= $\operatorname{argmax}\limits_{q(t_s),q(t_s+T),..,q(t_s+(d-2)T)} P[q(t_s), q(t_s+T), .., q(t_s+(d-1)T) = j', q(t_s+dT) = j, O(t_s), .., O(t_s+dT)|\lambda]$
for each user $k \in K$

1: Initialization (for each user $k \in K$):

$$\delta_0(j', j, k) = \pi_j.c_{j'j}(O(t_s)) \; and \; \psi_0(j', j, k) = 0 \; (m_j, m_{j'} \in M')$$

2: Recursive step (for each user $k \in K$):

$$\delta_d(j', j, k) = \max_{m_{j''} \in M'} [\delta_{d-1}(j'', j', k)a(j'', j', j)].c_{j'j}(O(t_s + dT))$$

$$\psi_d(j', j, k) = \operatorname*{argmax}_{m_{j''} \in M'} [\delta_{d-1}(j'', j', k)a(j'', j', j)]$$

where ($1 \leq d \leq d_{max}$ and $m_j, m_{j'}, m_{j''} \in M'$)

3: Termination (for each user $k \in K$):

$$P^*(k) = \max_{m_{j'}, m_j \in M'} \delta_{d_{max}}(j', j, k)$$

$$u^*_{d_{max}}(k) = \operatorname*{arg\,j}_{m_{j'}, m_j \in M'} max[\delta_{d_{max}}(j', j, k)] \;\; u^*_{d_{max}-1}(k) = \operatorname*{arg\,j'}_{m_{j'}, m_j \in M'} max[\delta_{d_{max}}(j', j, k)]$$

4: State sequence backtracking:

$$u^*_d(k) = \psi_{d+1}(u^*_{d+1}, u^*_{d+2}, k) \; d = (d_{max} - 2), (d_{max} - 3), ..., 0$$

5: Output (for each user $k \in K$): $\{q(t_s)$=$u^*_0(k)$, $q(t_s + T)$=$u^*_1(k)$, .., $q_i(t_s + dT)$=$u^*_d(k)..q_i(t_s + \tau.T)$=$u^*_\tau(k)\}$

Real-Time applicability of Adaptive-HMM: There were some constraints to directly using standard HMM model and Viterbi algorithm to our real-time application scenario. Regarding length of time window (say W) the standard Viterbi algorithm requires $O(W)$ operations. But the standard algorithm is not applicable in the case of a streamed input (with potentially no ending in sequence) and requirement of output within bounded delay. Regarding size of state space (say S), the standard Viterbi algorithm requires $O(S^2)$ operations, and still even on average $O(S\sqrt{S})$ operations by a modified version of Viterbi [64]. Thus for real-time applicability we have designed a model that is activity context aware with: (a) bounded length of time window (t_s to ($t_s + d_{max}.T$)), and (b) varying size of system state M' that is the set of motion activated nodes (in the time window) and their 1-hop or 2-hop neighbor nodes in $EATG$ (explained earlier). Therefore depending on the amount of activity in sensed binary motion data *Adaptive-HMM* dynamically selects the state space and HMM order.

6.2.2 Path Disambiguation Algorithm *CPDA*

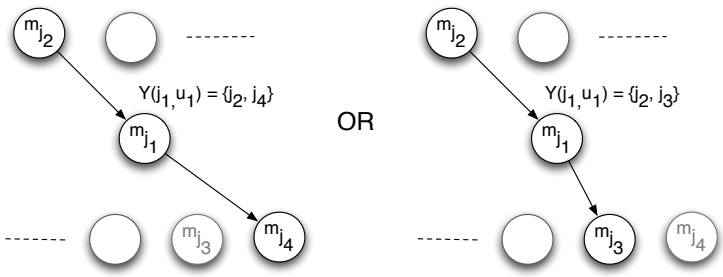


Figure 6.8 Explanation of *CPDA*. The *non-feature* node m_{j_1} has property $Y(j_1, u_1)$ (u_1 is the identifier of one of the users passing through m_{j_1}). $Y(j_1, u_1)$=$\{j_2, j_4\}$ is the scenario on left-hand side, and $Y(j_1, u_1)$=$\{j_2, j_3\}$ is the scenario on right-hand side.

The output state sequences from *Adaptive-HMM* in each time window (of length $d_{max}.T$) is partially disambiguated from the effect of path overlap or crossover. But it can't always remove longer term path ambiguity that spreads beyond the *Adaptive-HMM* time window. To alleviate this, *FindingHuMo* applies a proposed *Crossover Path Disambiguation Algorithm* or *CPDA* to the joint *Adaptive-HMM* output of last C number of time windows. It works as follows:

1. Combining *Adaptive-HMM* output of last C HMM time windows, it creates a directed graph $G_I = (V_I, E_I, P)$ (called *Interaction Graph*) that contains nodes V_I and edges E_I belonging to the output state sequences. P (=$\{P(j)\}$ $j \in V_I$) is set of node property, explained in next step.

2. In the constructed *Interaction Graph*, each node j is given a state variable $P(j)$ containing information about how the user paths from the incoming nodes are distributed into the outgoing nodes. $P(j) = \{Y(j, u)\}$ for every user u passing through node j. Nodes with deterministic and fixed value of $Y(j, u)$ (like nodes containing exactly one path, entry/exit nodes) are called *feature* nodes. The other intermediate nodes with possibly different values of $Y(j, u)$ are called *non-feature* nodes. Say on node m_{j_1}, one of the incoming node to node m_{j_1} is m_{j_2}, and one of the outgoing nodes from m_{j_1} is m_{j_3}. Then for user path u_i, if $Y(j_1, u_i) = \{j_2, j_3\}$, then it indicates that path of u_i goes from m_{j_2} through m_{j_1} to m_{j_3}.

3. Now based on the constraint on path distributions (imposed by G_I) and the defined state values $Y(j, u)$ of *feature nodes*, the system applies Bayesian Network Inference on G_I to calculate most desirable state values $Y(j, u)$ of non-feature nodes. The conditional probability table value is selected as follows: $P(Y(j_1, u_i) = \{j_2, j_3\}|j_2) = a''(m_{j_2}, m_{j_3})$ (thus utilizing $EATG$ graph).

4. Finally, if for some *non-feature node* the $Y(j, u)$ contradicts with the state sequence computed by *Adaptive-HMM*, path segments in the state sequences are switched to follow $Y(j, u)$.

Now we explain the proposed $CPDA$ algorithm step-by-step through a working example shown in Figure 6.7(b).

1. After computing $Adaptive-HMM$ for the last 2 time windows (thus here C=2) the combined output state sequences are: (i) 97 $\rightarrow$ 96 $\rightarrow$ 95 $\rightarrow$ 99 $\rightarrow$ 111 $\rightarrow$ 114, (ii) 114 $\rightarrow$ 114 $\rightarrow$ 111 $\rightarrow$ 99 $\rightarrow$ **111** $\rightarrow$ **110**, and (iii) 97 $\rightarrow$ 96 $\rightarrow$ 95 $\rightarrow$ 99 $\rightarrow$ **95** $\rightarrow$ **96**. These form the *Interaction Graph* $G_I = (V_I, E_I, P)$ containing the nodes and the corresponding edges, as shown in Figure 6.7(b).

2. Nodes 99 and 111 here are the *non-feature* nodes. Node 99 has 2 incoming nodes (95 and 111) and 2 outgoing nodes (111 and 95). Similarly node 111 has 1 incoming node (99) and 2 outgoing nodes (114 and 110). The for example for the user (say u_1) who was moving in the path $114 \rightarrow 111 \rightarrow 99$... can cause for node 99: $Y(99, u_1) = \{111, 111\}$ or $Y(99, u_1) = \{111, 95\}$.

3. Now after running the Bayesian Network Inference, for example for node 99 and for user u_1, $P(Y(99, u_1) = \{111, 95|111\}) > P(Y(99, u_1) = \{111, 111|111\})$.

4. Thus the path (...$111 \rightarrow 99 \rightarrow 95$...) suggested by $Y(99, u_1) = \{111, 95|111\}$ contradicts the part of $Adaptive - HMM$ output path ...$111 \rightarrow 99 \rightarrow 111$.... Therefore the path $114 \rightarrow 114 \rightarrow 111 \rightarrow 99 \rightarrow \mathbf{111} \rightarrow \mathbf{110}$ is corrected to $114 \rightarrow 114 \rightarrow 111 \rightarrow 99 \rightarrow \mathbf{95} \rightarrow \mathbf{96}$. This triggers the path $97 \rightarrow 96 \rightarrow 95 \rightarrow 99 \rightarrow \mathbf{95} \rightarrow \mathbf{96}$ corrected to $97 \rightarrow 96 \rightarrow 95 \rightarrow 99 \rightarrow \mathbf{111} \rightarrow \mathbf{110}$. This is because the path constraint has to be satisfied for each node that number of incoming user paths is equal to the number of outgoing user paths. In this way the path disambiguation is performed in $CPDA$.

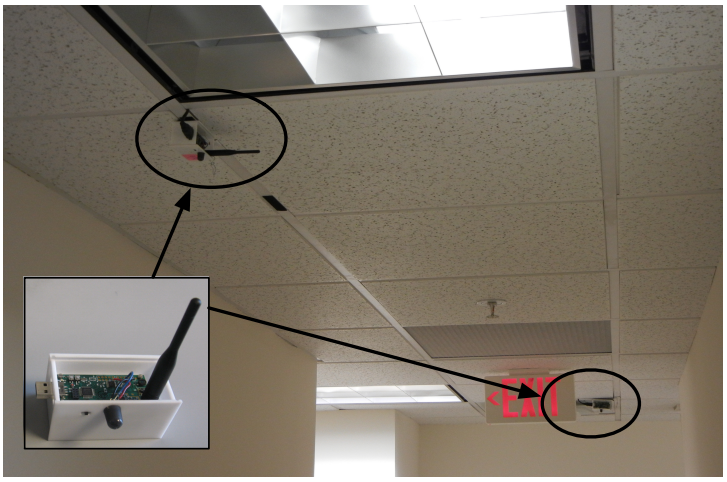
Therefore Bayesian inference in $CPDA$ helps eliminate some path ambiguity. This finishes the description of the proposed system $FindingHumo$.

6.3 Performance Evaluation

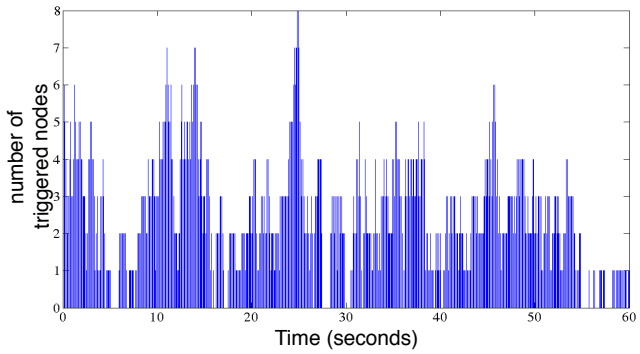
In this section we first explain our experimental setup in a real smart environment, followed by system performance analysis of multi-user tracking experiments.

6.3.1 System setup

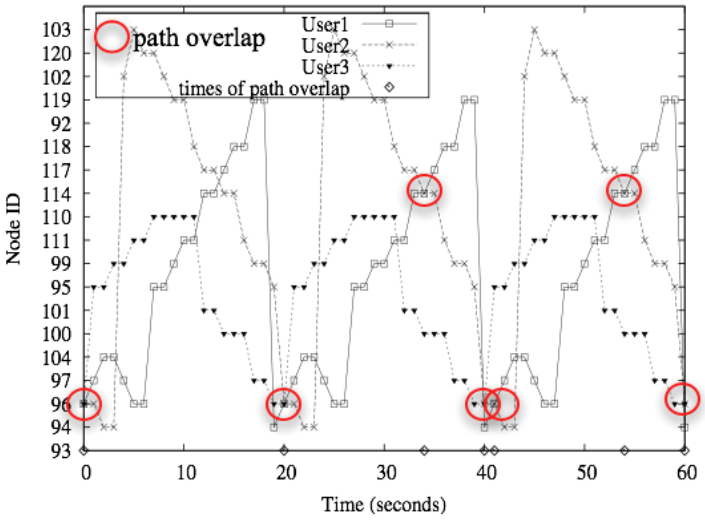
A network of 20 $TelosW$ [52] static wireless sensor nodes (Figure 6.9(a)) are deployed throughout the 30 meter x 30 meter floor (Figure 6.1) workplace environment (in workplace of Department of Computer Science, Georgia State University). As earlier shown in the physical layout in Figure 6.1, the sensor nodes are deployed mainly in the hallways, key positions (e.g. entry points: nodes 103, 104; exit points: nodes 97, 104; positions with high motion activities in



(a)



(b)



(c)

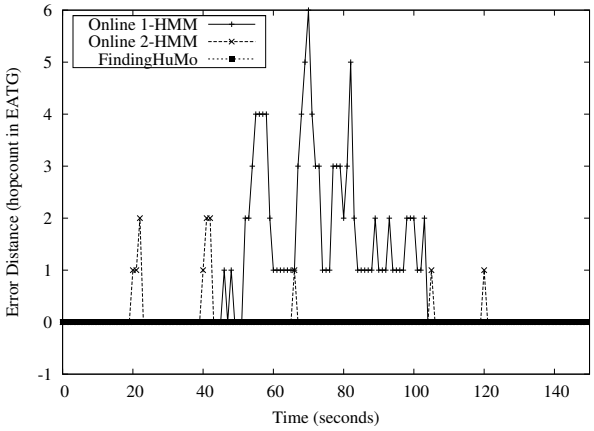
Figure 6.9 (a) Testbed deployment of PIR motion sensor nodes in a smart workplace environment. (b) Number of motion triggered nodes during 3 user experiment. (c) Ground truth motion trajectories of 3 users during experiment, with the path overlap/crossover shown.

workday: nodes 118, 117, 102,114, 100, 101), some rooms (e.g. printer room: node 92, kitchen: node 99, busy lab: node 93). These nodes are fixed on the ceiling and each of them are equipped with Panasonic AMN-31111 PIR (passive infrared) motion sensor. The $TelosW$ sensor nodes have sensor wake-on capability [52] to rationalize MCU (the processing unit) usage. The detected motion data (sampled at 10 Hz when event triggered by motion) are collected by the base station through multi-hop communication (formed a 5-hop network) and stored in a back-end database. The choice for system parameters are as follows. Length of unit timeslot T has been chosen as 1 second, M contains 20 motion sensor nodes, $d_{max}.T$ is 5 seconds. Based on system testing and evaluation, there were no false positive or false negative observed from the motion sensor. However, the tracking challenges come from factors like time synchronization, loss of transmitted sensor data, non-uniform node distribution and large number of overlapping users.

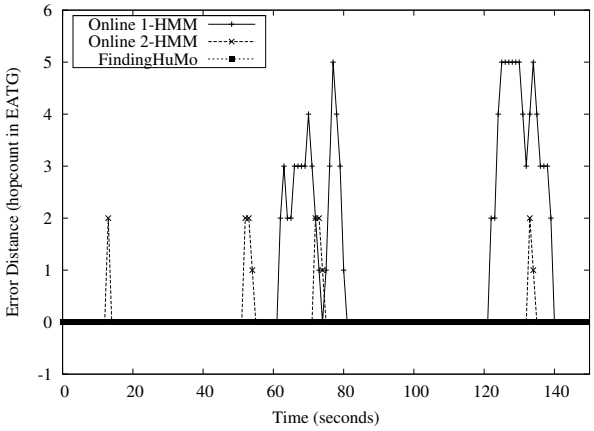
6.3.2 Multi-user Tracking Experiment

Experiment setup: In the performance evaluation experiment, three users (say $U1$, $U2$ and $U3$) are made to repeatedly move through the space in overlapping paths (as shown in Figure 6.2 and Figure 6.9(c)). The ground truth of user position was recorded by the moving user, to compare later with computed user trajectories. The ground truth has been compared to the following: (i) offline 1-HMM (first order HMM computed offline on full time data), (ii) offline full state 2-HMM (second order HMM computed offline on full time data), (iii) online full state 1-HMM (first order HMM computed online on time window of data), (iv) online full state 2-HMM (second order HMM computed online on time window of data), and (v) $FindingHumo$ (this uses second order HMM computed online on time window of data with activated subset of states, and then applying path disambiguation algorithm). It is important to mention that no current method was found in the existing literature, that fits this application scenario and requirements (binary sensor data, no geometric model etc.). $FindingHumo$ is compared here with different possible configurations of HMM computation, showing the utilities of: online time window based HMM, partial state HMM and path disambiguation algorithm.

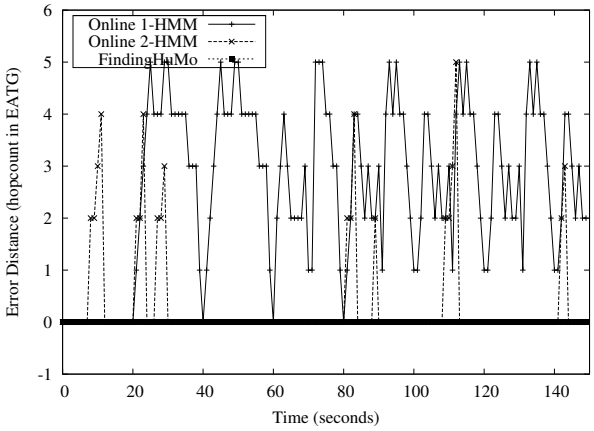
Performance evaluation of multi-user experiment: The 3 user overlapping path experiment



(a)



(b)



(c)

Figure 6.10 (a) Tracking error (hopcount in $EATG$, between ground truth node and detected node) measured for user1. (b) Tracking error measured for user2. (c) Tracking error measured for user3.

was conducted for about 150 seconds. Figure 6.9(b) shows the number of motion triggered nodes through 60 seconds run of the experiment process. This indicates that a lot of time more or less than 3 (user number) nodes are triggered at once. This is because of real scenarios: the different and non-uniform walking speeds of users, time synchronization issue (thus unreliable state sequences), the non-uniformity of node deployments etc. This creates the challenge for tracking algorithms to decode 3 best states indicating the state of each user. The tracking goal is to locate users with key logical points, instead of exact co-ordinate. Therefore the tracking error is measured with hopcount in the $EATG$ graph. Say the ground truth is state m_1 and detected state is m_2, then the tracking error is the shortest hopcount between nodes m_1 and m_2 in $EATG$. This is valid because adjacent nodes in $EATG$ indicate direct physical reachability. If the error had to be measured in terms of distance, then the error distance could be measured as the summation of all hop distances. This doesn't change the general applicability of our designed system.

Figures 6.10(a), 6.10(b) and 6.10(c) show the performance of $FindingHuMo$ for tracking user1, user2 and user3 respectively. There are some key observations made from the comparative performance analysis. (i) Full-state HMM computation time overhead is much higher than partial-state computation. (ii) Online 1-HMM and online 2-HMM (use same time windows as that of $FindingHumo$) performed same (same tracking error) as their corresponding offline version. Therefore it is validated that here time window based online computation in $FindingHumo$ does not affect the optimality of computed path. (iii) As shown in Figures 6.10(a), 6.10(b) and 6.10(c), 2-HMM has much lesser tracking errors than 1-HMM. This is the advantage of using second order HMM for overlapping paths. (iv) Finally, as in Figures 6.10(a), 6.10(b) and 6.10(c), $FindingHuMo$ showed no error in computed path. The errors that occurred in online 2-HMM (mostly 1 or 2 hop error distances) were locally corrected by path disambiguation algorithm in $FindingHuMo$. The average tracking error (in terms of tracking distance in hopcount) per unit timeslot T is as follows: (a) 1-HMM: 0.75 for user1, 0.78 for user2, 2.74 for user3; (b) 2-HMM: 0.08 for user1, 0.1 for user2, 0.35 for user3; (c) $FindingHuMo$: 0.00 for all users. Tracking User3 had more errors for just HMM based computation, because it had more crossover with other users. But overall $FindingHuMo$ system corrects this error for all users, by added inference in $CPDA$.

In addition to overlapping path experiment, also test was conducted with non-overlapping paths of those 3 users. The system performance indicated same trend as for the overlapping scenario. Therefore overall, the experimental results show that *FindingHumo* performs much better than other comparative configurations in computing accurate multi-user motion trajectories.

6.4 Summary

This work opens a relatively under-explored area where real-time multi-target tracking is done without any geometric model and with simple non-invasive sensors. This work presents a novel design of *FindingHuMo* (*Finding Hu*man *Mo*tion), a real-time user tracking system for Smart Environments. *FindingHuMo* can perform device-free tracking of multiple users in the Hallway Environments, just from non-invasive and anonymous (not user specific) binary motion sensor data stream. It can solve complex challenges in multi-user tracking where user motion trajectories may crossover with each other in all different ways. The performance improvement is demonstrated with results from experiments in real testbed in a smart workplace environment.

PART 7

CONCLUSIONS

This doctoral thesis in essence, explores human motion activity aware sensor networks protocol designs and applications for Smart Environments. Such activity-aware sensor networks system allows performance improvement of spatial and temporal data collection operations, and to save critical resources like node energy, network lifetime. The activity-aware application in this thesis explores non-invasive multi-user tracking with binary motion sensor network in Smart Workplace. Machine Learning based knowledge mining applied on collective spatial-temporal binary motion sensor data has led to fairly accurate tracking of multiple users' movement trajectories.

The first part of this thesis describes designing activity-aware sensor networks for Smart Environments in terms of three proposed protocols: *ActSee*, *EAR* and *Actisen*. *ActSee* [1] is an activity-aware radio duty cycling protocol, given the sensor network can use any routing protocol of it's choice. Then *EAR* [2] is an activity-aware and energy-balanced routing protocol, given the sensor network can use any radio duty cycling protocol. Finally the complete *ActiSen* [3] system is a complete sensor networking solution with activity-awareness integrated in all of: sensing, radio duty cycling and routing.

The second part of this thesis describes the works on an activity-aware sensor network application: real-time non-invasive tracking of multiple users' motion trajectories with binary motion sensor networks in Smart Workplace environment. The designed algorithm is called *FindingHuMo* or *Finding Human Motion*. *FindingHuMo* [4] can perform device-free tracking of multiple (unknown and variable number of) users in the Hallway Environments, just from non-invasive and anonymous (not user specific) binary motion sensor data stream.

REFERENCES

[1] S. Tang, D. De, W.-Z. Song, D. Cook, and S. Das, "ActSee: Activity-Aware Radio Duty-Cycling for Sensor Networks in Smart Environments," in *Eighth IEEE International Conference on Networked Sensing Systems (IEEE INSS)*, 2011.

[2] D. De, W.-Z. Song, S. Tang, and D. Cook, "EAR: An Energy and Activity Aware Routing Protocol for Wireless Sensor Networks in Smart Environments," *The Computer Journal*, vol. 55, no. 12, 2012.

[3] D. De, S. Tang, W.-Z. Song, D. Cook, and S. K. Das, "ActiSen: Activity-aware Sensor Network in Smart Environments," *Journal of Pervasive and Mobile Computing (PMC)*, Jan. 2012.

[4] D. De, W.-Z. Song, M. Xu, D. Cook, and X. Huo, "FindingHuMo: Real-Time Tracking of Motion Trajectories from Anonymous Binary Sensing in Smart Environments," in *The 32nd International Conference on Distributed Computing Systems (ICDCS'12)*, Jun. 2012.

[5] "CASAS Smart Home Project," http://ailab.wsu.edu/casas/.

[6] H. Lee, M. Wicke, B. Kusy, O. Gnawali, and L. Guibas, "Data Stashing: Energy-Efficient Information Delivery to Mobile Sinks through Trajectory Prediction," in *IPSN*, 2010.

[7] I. Constandache, X. Bao, M. Azizyan, and R. R. Choudhury, "Did You See Bob?: Human Localization using Mobile Phones," in *MobiCom*, 2010.

[8] A. Wood, J. Stankovic, G. Virone, L. Selavo, Z. He, Q. Cao, T. Doan, Y. Wu, L. Fang, and R. Stoleru, "Context-aware wireless sensor networks for assisted living and residential monitoring," *Special Issue IEEE Networks*, vol. 22, no. 4, pp. 26–33, Jul. 2008.

[9] D. O. Olguín, B. N. Waber, T. Kim, A. Mohan, K. Ara, and A. Pentland, "Sensible Organizations: Technology and Methodology for Automatically Measuring Organizational Behavior," *IEEE Transaction on Systems, Man, and Cybernetics*, vol. 39, Feb. 2009.

[10] D. Lymberopoulos, T. Teixeira, and A. Savvides, "The BehaviorScope Framework for Enabling Ambient Assisted Living," *International Journal on Personal and Ubiquitous Computing*, vol. 14, no. 6, 2010.

[11] V. Srinivasan, J. Stankovic, and KaminWhitehouse, "Protecting your Daily In-Home Activity Information from a Wireless Snooping Attack," in *UbiComp*, 2008.

[12] M. Buettner, R. Prasad, M. Philipose, and DavidWetherall, "Recognizing Daily Activities with RFID-Based Sensors," in *UbiComp*, 2009.

[13] "Smart Home-based Health Platform for Behavioral Monitoring and Alteration for Diabetic and Obese Individuals," http://www.icta.ufl.edu/projects_nih.htm.

[14] "MavHome," http://ailab.uta.edu/mavhome/.

[15] "Smart Environments," http://wsnl.stanford.edu/smartenv.html.

[16] "Smart Medical Home," http://www.futurehealth.rochester.edu/smart_home/.

[17] "Spinner," http://www.media.mit.edu/resenv/spinner/introduction.html.

[18] S. Ahn and D. Kim, "Proactive Context-Aware Sensor Networks," in *EWSN*, 2006.

[19] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," in *21st Conference of the IEEE Computer and Communications Societies (INFOCOM)*, Jun. 2002.

[20] T. van Dam and K. Langendoen, "An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks," in *The 1st ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2003.

[21] J. Polastre, J. Hill, and D. Culler, "Versatile Low Power Media Access for Wireless Sensor Networks," in *The 2nd ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2004.

[22] M. Buettner, G. V. Yee, E. Anderson, and R. Han, "X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks," 2006.

[23] E. Gelenbe and E.-H. Ngai, "Adaptive qos routing for significant events in wireless sensor networks," in *Mobile Ad Hoc and Sensor Systems, 2008. MASS 2008. 5th IEEE International Conference on*, Sept 2008, pp. 410–415.

[24] J. H. Chang and L. Tassiulas, "Energy Conserving Routing in Wireless Ad-hoc Networks," in *INFOCOM*, 2000, pp. 22–31.

[25] ——, "Fast Approximate Algorithms for Maximum Lifetime Routing in Wireless Ad-hoc Networks," in *Networking*, 2000.

[26] Q. Li, J. Aslam, and D. Rus, "Online power-aware routing in wireless Ad-hoc networks," in *Mobicom*, 2001.

[27] Y. Xue, Y. Cui, and K. Nahrstedt, "A Utility-based Distributed Maximum Lifetime Routing Algorithm forWirelessNetworks," in *QShine*, 2005.

[28] E. Gelenbe and R. Lent, "Power-aware ad hoc cognitive packet networks," *Ad Hoc Networks*, vol. 2, no. 3, pp. 205 – 216, 2004, quality of service in ad hoc networks. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1570870504000241

[29] E. Gelenbe and E. C. H. Ngai, "Adaptive random re-routing for differentiated qos in sensor networks." in *BCS Int. Acad. Conf.*, E. Gelenbe, S. Abramsky, and V. Sassone, Eds. British Computer Society, 2008, pp. 343–354. [Online]. Available: http://dblp.uni-trier.de/db/conf/bcs/bcs2008.html#GelenbeN08

[30] E. Gelenbe and C. Morfopoulou, "Power savings in packet networks via optimised routing," *Mob. Netw. Appl.*, vol. 17, no. 1, pp. 152–159, Feb. 2012. [Online]. Available: http://dx.doi.org/10.1007/s11036-011-0344-0

[31] T. Mahmoodi, "Energy-aware routing in the cognitive packet network," *Perform. Eval.*, vol. 68, no. 4, pp. 338–346, Apr. 2011. [Online]. Available: http://dx.doi.org/10.1016/j.peva.2011.02.002

[32] E. Gelenbe, "Steps toward self-aware networks," *Commun. ACM*, vol. 52, no. 7, pp. 66–75, Jul. 2009. [Online]. Available: http://doi.acm.org/10.1145/1538788.1538809

[33] S. Dobson, S. Denazis, A. Fernández, D. Gaïti, E. Gelenbe, F. Massacci, P. Nixon, F. Saffre, N. Schmidt, and F. Zambonelli, "A survey of autonomic communications," *ACM Trans. Auton. Adapt. Syst.*, vol. 1, no. 2, pp. 223–259, Dec. 2006. [Online]. Available: http://doi.acm.org/10.1145/1186778.1186782

[34] K. Kar, M. Kodialam, T. V. Lakshman, L. Tassiulas, and R. Tassiulas, "Routing for Network Capacity Maximization in Energy-constrained Ad-hoc Networks," in *INFOCOM*, 2003.

[35] L. Lin, N. B. Shroff, and R. Srikant, "Asymptotically Optimal Energy-Aware Routing for Multihop Wireless Networks With Renewable Energy Sources," *IEEE/ACM Transactions on Networking*, vol. 15, Oct. 2007.

[36] C. Ok, P. Mitra, S. Lee, and S. Kumara, "Maximum Energy Welfare Routing in Wireless Sensor Networks," in *NETWORKING 2007*, 2007.

[37] R. Menchaca-Mendez and J. J. Garcia-Luna-Aceves, "Robust and Scalable Integrated Routing inMANETs Using Context-Aware Ordered Meshes," in *INFOCOM*, 2010.

[38] L. R. Rabinder, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, Feb. 1989.

[39] J. Shin, L. J. Guibas, and F. Zhao, "A distributed algorithm for managing multi-target identities in wireless ad-hoc sensor networks," in *Proceedings of the 2nd international conference on Information processing in sensor networks (IPSN'03)*, Palo Alto, California, USA, Apr. 2003.

[40] Z. Zhong, T. Zhu, D. Wang, and T. He, "Tracking with Unreliable Node Sequences," in *Proceedings of the 28th Conference on Computer Communications (INFOCOM'09)*, Reo de Janeiro, Brazil, Apr. 2009, pp. 1215–1223.

[41] D. Zhang, Y. Liu, and L. Ni, "RASS: A Real-time, Accurate and Scalable System for Tracking Transceiver-free Objects," in *Ninth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom'11)*, Seattle, USA, 2011.

[42] X. Chen, A. Edelstein, Y. Li, M. Coates, M. Rabbat, and A. Men, "Sequential Monte Carlo for Simultaneous Passive Device-Free Tracking and Sensor Localization Using Received Signal Strength Measurements," in *International Conference on Information Processing in Sensor Networks (IPSN'11)*, Chicago, USA, 2011.

[43] Z. Wang, E. Bulut, and B. K. Szymanski, "Distributed Target Tracking with Directional Binary Sensor Networks," in *GLOBECOM*, 2009.

[44] J. Singh, R. Kumar, U. Madhow, S. Suri, and R. Cagley, "Tracking Multiple Targets Using Binary Proximity Sensors," in *Proceedings of the 6th international conference on Information processing in sensor networks (IPSN'07)*, Cambridge, Massachusetts, Apr. 2007.

[45] M. Han, W. Xu, H. Tao, and Y. Gong, "An Algorithm for Multiple Object Trajectory Tracking," in *Conference on Computer Vision and Pattern Recognition (CVPR'04)*, Washington DC, USA, Jun. 2004.

[46] H. Ardö, K. Åström, and R. Berthilsson, "Real Time Viterbi Optimization of Hidden Markov Models for Multi Target Tracking," in *WMVC*, 2007.

[47] P. Nillius, J. Sullivan, and S. Carlsson, "Multi-Target Tracking - Linking Identities using Bayesian Network Inference," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2, 2006, pp. 2187–2194. [Online]. Available: http://dx.doi.org/10.1109/cvpr.2006.198

[48] B. Ristic, S. Arulampalam, and N. Gordon, *Beyond the Kalman Filter: Particle Filters for Tracking Applications*, A. House, Ed., 2004.

[49] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Transactions of the ASME Journal of Basic Engineering*, no. 82 (Series D), pp. 35–45, 1960. [Online]. Available: http://www.cs.unc.edu/\~{}welch/kalman/media/pdf/Kalman1960.pdf

[50] H. Ardö, K. Åström, and R. Berthilsson, "Online Viterbi Optimisation for Simple Event Detection in Video," in *ICVSS*, 2007.

[51] X. Ma, D. Schonfeld, and A. Khokhar, "Distributed multi-dimensional hidden Markov model: theory and application in multiple-object trajectory classification and recognition," in *Conference of Multimedia Content Access: Algorithms and Systems II (SPIE'08)*, San Jose, CA, USA, Jan. 2008.

[52] G. Lu, D. De, M. Xu, W.-Z. Song, and B. Shirazi, "TelosW: Enabling Ultra-Low Power Wake-On Sensor Network," in *Seventh International Conference on Networked Sensing Systems (INSS'10)*, Kassel, Germany, Jun. 2010.

[53] N. L. Philip Levis, *TOSSIM: A Simulator for TinyOS Networks*, Jun. 2003.

[54] P. Basu and J. Redi, "Effect of Overhearing Transmissions on Energy Efficiency in Dense Sensor Networks," in *IPSN*, 2004.

[55] A. B. Atkinson, "On the measurement of inequality," *Journal of Economics Theory*, 1970.

[56] S. Lin, J. Zhang, G. Zhou, L. Gu, J. A. Stankovic, and T. He, "ATPC: Adaptive Transmission Power Control for Wireless Sensor Networks," in *Proceedings of the 4th international conference on embedded networked sensor systems (SenSys '06)*. New York, NY, USA: ACM Press, 2006, pp. 223–236. [Online]. Available: http://dx.doi.org/10.1145/1182807.1182830

[57] T. Das and A. Parikh, "Statistical interpretation, decomposition and properties of Atkinson's inequality index," *Decisions in Economics and Finance*, 1982.

[58] G. Werner-Allen, P. Swieskowski, and M. Welsh, "MoteLab: a wireless sensor network testbed," in *Fourth International Symposium on Information Processing in Sensor Networks (IPSN)*, 2005.

[59] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection Tree Protocol," in *Proc. of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2009.

[60] Y. Peng, W. Song, R. Huang, M. Xu, and B. Shirazi, "Cascades: A Reliable Dissemination Protocol for Data Collection Sensor Network," in *IEEE Aerospace Conference*, Big Sky, MT, USA, Mar. 2009.

[61] H. Lin, M. Lu, N. Milosavljevic, and J. Gao, "Composable Information Gradients in Wireless Sensor Networks," in *IPSN 2008*, 2008.

[62] B. Kusy, H. Lee, M. Wicke, N. Milosavljevic, and L. Guibas, "Predictive QoS Routing to Mobile Sinks in Wireless Sensor Networks," in *IPSN 2009*, 2009.

[63] S. M. Thede and M. P. Harper, "A second-order Hidden Markov Model for part-of-speech tagging," in *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics (ACL'99)*, Stroudsburg, PA, USA, 1999.

[64] S. Patel, "A lower-complexity Viterbi algorithm," in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP'95)*, Detroit, USA, 1995.